Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Streamování videa pomocí multicastu v IPv6 sítích založených na open source routovacích platformách

Multicast Video Streaming in IPv6 Networks based on Open Source Routing Platform

2019 **Bc. Antonín Škrobánek**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Antonín Škrobánek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Streamování videa pomocí multicastu v IPv6 sítích založených na open source routovacích platformách.
Multicast Video Streaming in IPv6 Networks based on Open Source Routing Platform.

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je navrhnout otevřené řešení pro streamování videa pomocí multicastu v prostředí IPv6 sítí. Jedním z řešení je možnost využití oteřeného řešení platformy XORP.

Řešení práce musí obsahovat následující body:
1. Studium a popis IPv4 a IPv6 multicastu.
2. Studium a popis audiokodeků a videokodeků.
3. Výběr a konfigurace vhodných linuxových programů.
4. Konfigurace linuxových zařízení pomocí bash skriptů.
5. Zátěžové testování v laboratorních podmínkách.

Seznam doporučené odborné literatury:

[1]Minoli, D. *Linear and Non-Linear Video and TV Applications: Using IPv6 and IPv6 Multicast.* Wiley 2012
Sze, V., Budagavi, M., Sullivan, G.J. *High Efficiency Video Coding (HEVC): Algorithms and Architectures (Integrated Circuits and Systems).* Springer 2014

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Pavel Nevlud**

Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2019

prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *14. dubna 2019*

podpis studenta

Poděkování

Rád bych poděkoval vedoucímu diplomové práce Ing. Pavlu Nevludovi za odbornou pomoc a konzultaci při vytváření této diplomové práce.

Abstrakt

Diplomová práce je orientovaná na navržení otevřeného řešení pro streamování videí pomocí multicastu v IPv6 sítích založené na open source routovacích platformách a programech. První polovina práce je věnována teoretické části rozdělené do dvou kapitol. V první kapitole je rozebrána problematika multicastu, jeho distribuce, adresace a protokoly které využívá v různých sítích. Druhá kapitola je zaměřena na popis kodeků sloužících pro zpracování audio a video dat. Obsahuje popis samotných kodeků, jejich rozdělení a princip. Zbylé tři části jsou věnovány praktickému ověření v laboratorních podmínkách na vlastně vytvořené topologii sítě. Tyto části obsahují konfigurace linuxových programů, linuxových zařízení pomocí bash skriptů a zátěžové testování.

Klíčová slova

Multicast; IGMP; MLD; PIM; Kodek; Bash; Xorp; MRD6; SMCRoute

Abstract

The thesis is focused on designing an open video streaming solution using multicast in IPv6 networks based on open source routing platforms and programs. The first half of the thesis is devoted to the theoretical part, divided into two parts. The first part deals with multicast, its distribution, addressing and protocols used in various networks. The second part is focused on the description of codecs used for processing audio and video data. It contains description of codecs themselves, their distribution and principle. The remaining three parts are devoted to practical verification in laboratory conditions on actually created network topology. These sections include configurations of Linux programs, Linux devices using bash scripts, and stress testing.

Key words

Multicast; IGMP; MLD; PIM; Codec; Bash; Xorp; MRD6; SMCRoute

Seznam použitých zkratek

Zkratka	Význam
AAC	Advanced Audio Coding
AC3	Audio Coding-3
ALAC	Apple Lossless Audio Codec
ALE	Apple Lossless Encoder
AppArmor	Application Armor
AVC	Advanced Video Coding
avi	Audio Video Interleave
BIDIR-PIM	Bidirectional Protocol Independent Multicast
BSD	Berkeley Software Distribution
BGP	Border Gateway Protocol
CD	Compact Disc
dst	Destination
DVB-T	Digital Video Broadcasting-Terrestrial
DVB-T2	Digital Video Broadcasting-Terrestrial 2
DVD	Digital Versatile Disc
DVMPR	Distance Vector Multicast Routing Protocol
FLAC	Free Lossless Audio Codec
GB	Gigabyte
HD	High Definition
HDTV	High Definition TV
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
ICON	ICSI Center for Open Networking
IGMP	Internet Group Management Protocol
IGMPv2	Internet Group Management Protocol Version 2
IGMPv3	Internet Group Management Protocol Version 3

IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
kbit/s	Kilobit za sekundu
kHz	Kilohertz
JVT	Joint Video Team
LAN	Local Area Network
LG	Leave Group
LXC	Linux Containers
MAAS	Metal As A Service
Mac OS	Macintosh Operating System
MAN	Metropol Area Network
Mbit/s	Megabit za sekundu
MLD	Multicast Listener Discovery
MLDv1	Multicast Listener Discovery Version 1
MLDv2	Multicast Listener Discovery Version 2
MMS	Microsoft Media Server
MP1	MPEG-1 Audio Layer 1
MP2	MPEG-2 Audio Layer 2
MP3	MPEG Audio Layer 3
MPEG	Moving Picture Experts Group
MPEG-1	Moving Picture Experts Group 1
MPEG-2	Moving Picture Experts Group 2
MPEG-3	Moving Picture Experts Group 3
MPEG-4	Moving Picture Experts Group 4
mp1v	MPEG-1 Video
mp2v	MPEG-2 Video
mp4a	MPEG-4 Audio
mp4v	MPEG-4 Video
mpga	MPEG Audio

MQ	Membership Queries
MR	Membership Reports
MRD6	IPv6 Multicast Routing Daemon
MTU	Maximum Transmission unit
mux	Muxer
OSPF	Open Shortest Path First
OSPFv2	Open Shortest Path First Version 2
OSPFv3	Open Shortest Path First Version 3
PAN	Personal Area Network
PIM	Protocol Independent Multicast
PIM-DM	Protocol Independent Multicast Dense Mode
PIM-SSM	Protocol Independent Multicast Source-Specific Multicast
PIM-SM	Protocol Independent Multicast Sparse Mode
ps	Program Stream
RAM	Random Access Memory
RIID	RP Interface Identifier
RIP	Routing Information Protocol
RP	Rendezvous Point
SELinux	Security-Enhanced Linux
SM	Sparse Mode
SMCRoute	Static Multicast Routing Daemon
SSM	Source-Specific Multicast
std	Standard
TCP	Transmission Control Protocol
ttl	Time to live
UDP	User Datagram Protocol
UHD	Ultra High Definition
USA	United States of America
VCEG	Video Coding Experts Group
VLC	VLC Media Player

VRML	Virtual Reality Modeling Language
WAN	Wide Area Network
WMA	Windows Media Audio
WMV	Windows Media Video
WMV1	Windows Media Video 7
WMV2	Windows Media Video 8
WMV3	Windows Media Video 9

Seznam ilustrací a seznam tabulek

Obsah

Úvod

V digitálním prostředí a době internetu je velmi rozmanitá možnost komunikace mezi uživateli, kteří využívají mnoho aplikací a softwarů k její realizaci. Tuto komunikaci je ovšem zapotřebí zprostředkovat a docílit co možná nejefektivnějšího způsobu její distribuce skrze počítačovou síť. K efektivní distribuci slouží speciální druh komunikace známý jako multicast. Jedná se o jeden ze čtyř druhů komunikace, a díky jeho účinnosti se řadí mezi nejvýznamnější způsoby, jak distribuovat data. Není proto velkým překvapením, že se tohoto způsobu distribuce hojně využívá i v oblasti streamování audio a video obsahu. V praxi je multicast využíván např. v oblasti televizního vysílání známé jako IPTV nebo videokonferencí.

Cílem diplomové práce je navrhnout otevřené řešení pro streamování videí pomocí multicastu v prostředí IPv6 sítí založené na open source routovacích platformách. Těchto routovacích platforem existuje celá škála a nabízí tak uživatelům více možností, jak multicastový přenos realizovat. Mezi programy, které dokáží multicastový přenos realizovat patří např. Xorp, MRD6 nebo SMCRoute.

V první kapitole teoretické části práce je zahrnuta veškerá problematika z oblasti distribuce multicastu skrze počítačové sítě a samotný popis multicastového přenosu, který lze realizovat v IPv4 a IPv6 sítích. Na počátku kapitoly je popsána všeobecná počítačová síť, její výhody a základní rozdělení dle geografické rozlohy. Následně je popis věnován samotnému multiticastu, kde je rozebrán jeho princip a také zbylé druhy komunikace jejichž pojetí distribuce dat je odlišné. Závěr první části teorie je zaměřen na problematiku multicastu realizovaného v sítích IPv4, IPv6 a protokolu PIM. Jelikož se jedná o rozdílné sítě, je zde popsán způsob adresace v jednotlivých sítích a také využívané protokoly, které multicastu slouží k jeho realizaci. Co se týče protokolu PIM, pak tento protokol je společný pro oba druhy sítí a hlavním cílem je popis přenosových režimů, které protokol nabízí.

Druhá kapitola teoretické části je zaměřena na téma využití kodeků. K tomuto tématu je zahrnuta veškerá teorie, týkající se kodeků ať už z pohledu audio nebo video dat. Je zde popsáno, co si lze představit pod pojmem kodek, dále pak jejich základní rozdělení a samotný popis vybraných druhů s kterými je možné se v praxi setkat.

Třetí a čtvrtá část práce je věnována výběru a konfiguraci vhodných linuxových programů, které slouží pro realizaci multicastového přenosu a konfiguraci linuxových zařízení pomocí bash skriptů. Je zde zahrnut základní popis programů, jejich možnosti nastavení a konfigurace, které přispívají ke správně funkčnosti. Mezi tyto linuxové programy jsou vybrány programy Xorp, MRD6, SMCRoute, LXD a multimediální VLC Media Player. Konfigurace zařízení pomocí bash skriptů zahrnuje čtvrtou část práce a opětovně obsahuje jejich základní popis a ukázky vlastních skriptů, které slouží v praktické části.

Poslední kapitola diplomové práce je určena k reálnému testování v laboratorních podmínkách. Součástí kapitoly jsou výsledky testování programů Xorp, MRD6 a SMCRoute obohacené o výpisy nebo ukázky konfigurací na vlastích síťových topologiích.

1 Multicast IPv4 a IPv6

1.1 Počítačové sítě, jejich výhody a rozdělení

Počítačové sítě z anglického názvu computer networks byly vytvořeny v polovině dvacátého století. Od této doby prošly sítě podstatným technologickým rozvojem a dnes již nenalezneme téměř žádné zařízení, které není její součástí. Osobní počítače a celkově veškerá dnes již běžně dostupná technologie jako jsou chytré mobilní telefony, rozmanité softwary a mnoho dalších se neobjevují pouze v rozsáhlých podnicích, jak tomu v minulosti bývalo, ale také v běžné domácnosti. Tento fakt je dán také tím, že technologie se stávají levnějšími, a tak i dostupnějšími pro běžné uživatele. [8][9]

Obecně lze počítačovou síť popsat jako soustavu počítačů a zařízení, které dokáží mezi sebou vzájemně komunikovat a vyměňovat si informace. Nezbytnou podmínkou ovšem je, aby soustavu tvořili alespoň dva a více počítačů či zařízení, které jsou komunikace a výměny schopné. [3]

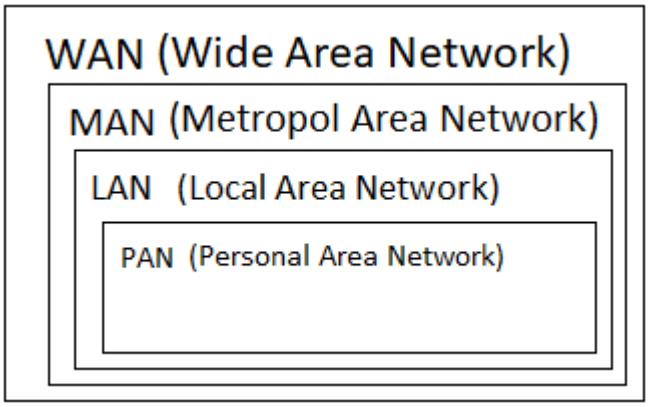
1.1.1 Výhody počítačové sítě

- Sdílení dat,
- sdílení hardwaru,
- přenos dat,
- ochrana dat,
- komunikace v síti. [3]

Sdílení, přenos dat a komunikace v síti vyplývají již ze samotné podstaty počítačové sítě. Ovšem jednou z podstatných výhod sítě je sdílení nejen dat, ale také hardwaru jako jsou tiskárny, skenery či síťové prvky, které mohou být využity pro společnou práci. V neposlední řadě je zde také ochrana dat. Ochrana spočívá v možnosti uchovávat data na jednom konkrétním místě v síti a typickým nositelem těchto dat je tzv. server. Serverem bývá ve většině případů počítač, který obsahuje utajované informace a tyto informace může distribuovat pouze vybraným uživatelům. [3]

1.1.2 Rozdělení sítí podle rozlohy

Možností, jak jednotlivé sítě klasifikovat je v dnešní době celá řada. Lze je rozdělit například podle přepojování, druhů přenášených signálů, vlastnictví apod. Nicméně nejtypičtějším rozdělením, které je využíváno, je dle rozlohy a v sítích se využívají tři základní označení. K tomuto označení se řadí tzv. lokální, metropolitní a rozlehlá síť nesoucí zkrácené označení LAN (Local Area Network), MAN (Metropol Area Network) a WAN (Wide Area Network). Často bývá k sítím řazena také síť označována jako PAN (Personal Area Network) neboli personální síť. Všechny sítě jsou koncipovány tak, že síť s větší rozlohou je složená z mnoha menších sítí, které tvoří souhrnný celek, a právě tento celek je označován za jednu síť z výše uvedených druhů. Pro lepší pojetí si lze představit, že jedna WAN síť je složená z několika MAN sítí, dále pak MAN síť z několika LAN sítí atd. [8][9][10]



Obrázek 1.1: *Rozdělení sítí podle rozlohy*

1.1.3 PAN (Personal Area Network)

Personální síť je druh, který se svou rozlohou řadí mezi nejmenší a nedá se tudíž očekávat, že jejich dosah bude větší než několik metrů. PAN síť i přes takto malé dosahy našla své uplatnění a slouží spíše pro jednotlivce či menší skupinu uživatelů k propojení různých druhů zařízení jako jsou mobilní telefony nebo počítače. Příkladná technologie, kterou lze do této kategorie řadit je technologie Bluetooth, ZigBee a Wifi. [9][10]

1.1.4 LAN (Local Area Network)

Lokální sítě v porovnání se sítěmi personálními dosahují poněkud větších vzdáleností. Nejedná se již o jednotlivce či menší skupinu, ale o síť jejíž rozloha se přiřazuje k určitému objektu neboli jednomu lokálnímu místu, které svou rozlohou čítá několik desítek a v krajním případě i stovek metrů. Takto rozsáhlými komplexy mohou být například školy a v neposlední řadě také firemní budovy, které jsou spravovány jedním či několika tzv. správci sítě neboli administrátory. V dnešní době zahrnuje LAN síť převážně dvě technologie, které se využívají ve většině případů. Těmito technologiemi je myšlena technologie Wifi a Ethernet. Lokální síť může být vytvořena jako síť, která pracuje samostatně a propojovat tak velké množství zařízení. Dnes jsou ve většině případů LAN sítě připojeny do internetu a poskytují uživatelům mnoho služeb, jako jsou: [8][10]

- Zasílání zpráv, email,
- využívání síťových zařízení,
- sdílení dat a aplikací. [10]

1.1.5 MAN (Metropol Area Network)

Metropolitní síť je síť, která se svou rozlohou řadí nad sítě lokální. U tohoto druhu dosahujeme vzdáleností nikoliv stovek metrů ale vzdáleností dosahující několika kilometrů, čímž roste také rozloha, kterou síť pokrývá. Zatím co u lokálních sítí je rozloha omezena v rámci budovy či většího komplexu, zde se jedná o poněkud významnější část rozlohy, a to v rámci města. [8][10]

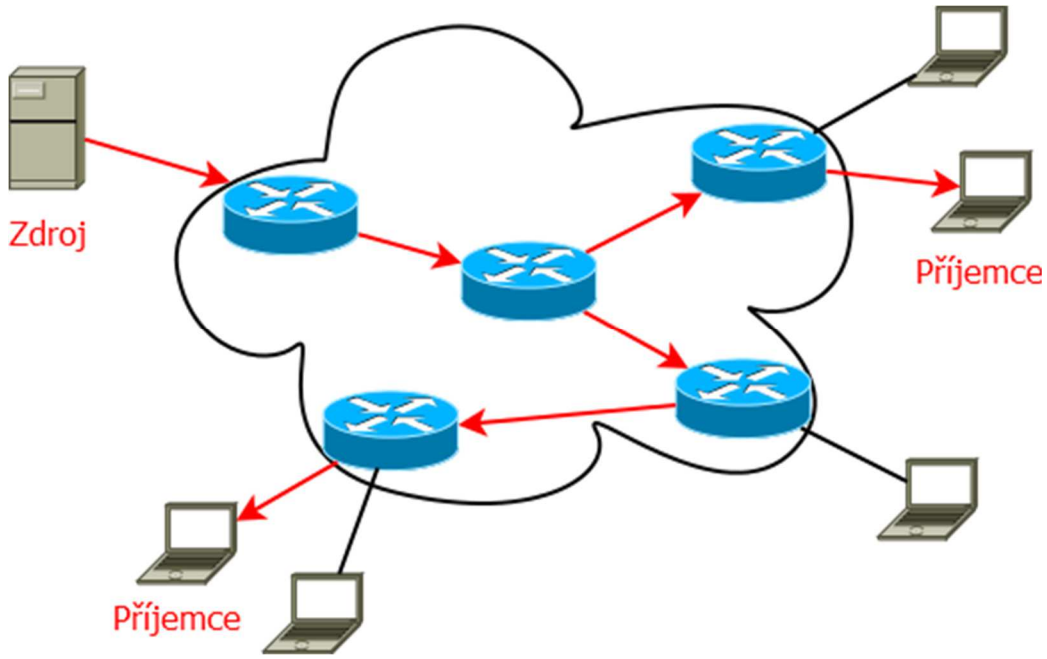
1.1.6 WAN (Wide Area Network)

Rozlehlá síť neboli WAN je komunikační síť pokrývající rozsáhlé území. Nejznámější WAN sítí je dnes již dobře známý Internet a rozlohou se jedná o největší síť pokrývající země, a dokonce i kontinenty. Sítě tohoto druhu jsou tvořeny několika tzv. řídicími počítači, které pro vzájemnou komunikaci využívají komunikační podsíť tvořenou speciálními datovými spoji. Datové spoje jsou ve většině případů ve vlastnictví organizací zabývající se službami v oblasti telekomunikací a jde především o datové spoje tvořené optickými kabely případně telefonní linkou. [8][10]

1.2 IP Multicast

Počítačové sítě, které jsou v dnešní době využívány, přinášejí pro uživatele mnoho výhod a jednou z nich je právě komunikace nebo posílání dat mezi uživateli. Důležitou podmínkou nebo také kritériem komunikace je její efektivita, se kterou se pojí právě multicastová komunikace.

Komunikace s využitím multicastu je velmi efektivní formou a jedná se o jeden ze čtyř druhů, které lze využít pro přenos dat skrze počítačovou síť k cílovým uživatelům. Multicast lze nahradit pojmem skupinové vysílání z čehož plyne, že cílem přenášených dat není jednotlivec, ale právě skupina uživatelů vyžadují totožné informace, které pocházejí z jediného zdroje a distribuují se ve stejný čas. Právě díky skutečnosti, že příjemcem není jeden uživatel, ale je těchto uživatelů více dopomáhá k tomu, že nedochází k zbytečnému zatěžování síťové infrastruktury a tím i zvýšení efektivity. Výhoda spočívá v tom, že nedochází k vytváření totožných dat již na cestě od samotného zdroje, ale až v místě, kde se rozvětvují samotní uživatelé. Multicastovou formu přenosu v hojné míře využívá například technologie IPTV pro distribuci digitálních služeb skrze počítačovou síť s využitím IP protokolu. [11][12][13]



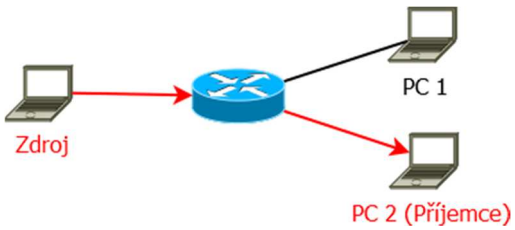
Obrázek 1.2: *IP Multicast*

1.3 Komunikace nevyužívající skupinové vysílání

Vyjma skupinového vysílání existují další dvě komunikace jejichž distribuce dat není založena na skupinách. Mezi tyto komunikace řadíme:

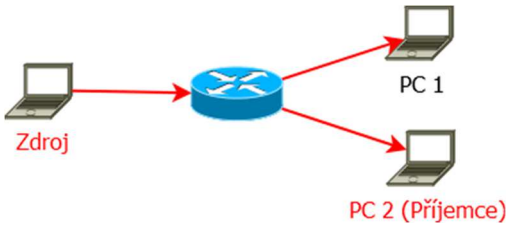
- Unicast,
- broadcast. [12]

Zatím co distribuce dat u multicastu je daleko sofistikovanější, než u těchto dvou druhů komunikace, i tyto našli své uplatnění v oblasti počítačových sítí. Nejjednodušší formu distribuce v oblasti sítí vykazují právě unicast a broadcast. Prvně zmiňovaný unicast je formou komunikace využívající velmi jednoduchý princip přenosu. Přenos spočívá v komunikaci jednoho zdroje s jediným příjemcem, při které jsou oba jasně identifikováni svou IP adresou. V porovnání s přenosem dat pomocí multicastu je tento přenos podstatně jednodušší, ale daleko méně efektivnější právě díky přímé komunikace zdroje pouze s jediným příjemcem. Pokud by jiný uživatel vyžadoval data z daného zdroje, dochází k vytváření dalšího spojení skrze počítačovou síť a tím i k zatěžování síťové infrastruktury. [12][14]



Obrázek 1.3: *Unicast*

U broadcastové komunikace je skutečnost zcela odlišná než u unicastu. Broadcast je zahrnut v jakékoliv LAN sítí a součástí komunikace je opětovně jeden zdroj. Tento zdroj distribuuje data nikoliv k jedinému příjemci, ale všem, kteří jsou součástí daného segmentu sítě. Broadcast lze využívat také pro komunikaci mezi síťovými prvky, pomocí kterého se objevují a přeposílají informace ostatním. [12][15]



Obrázek 1.4: *Broadcast*

1.4 Anycast

Anycastová komunikace je v podstatě postavena na principu multicastu. Anycast ovšem pracuje trochu odlišným způsobem, a proto nelze tuto komunikaci úplně zařazovat do druhů, které využívají skupinové vysílání. U anycastu jsou podobně jako u multicastu odeslána data na známou skupinu uživatelů, ale v konečné fázi distribuce, nejsou data rozkopírována všem. Data se odesílají pouze uživateli, který se v dané skupině nachází nejblíže. [16]

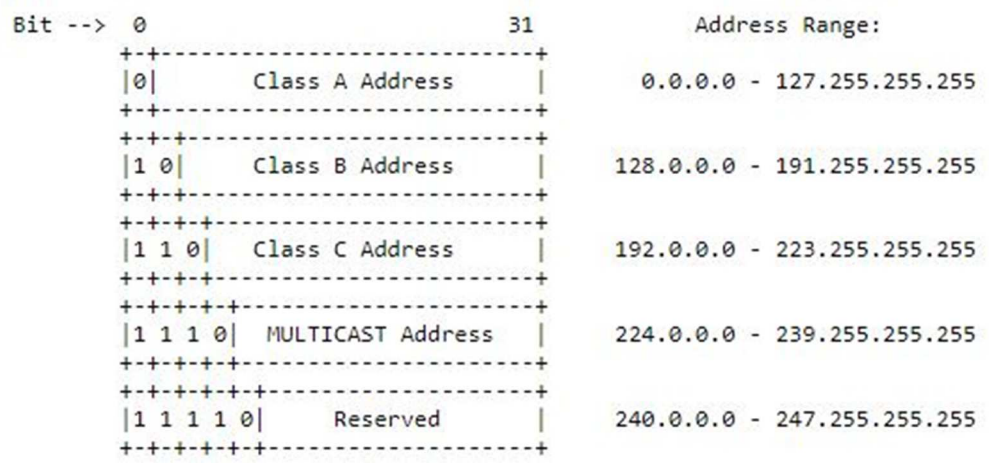
1.5 Multicast v IPv4

Z pohledu referenčního modelu ISO/OSI, který se skládá z celkem sedmi vrstev je pro multicastovou komunikaci nejvýznamnější třetí tzv. síťová vrstva. Na každé z těchto sedmi vrstev se nachází protokoly, které daným vrstvám přísluší.

Síťová vrstva má z pohledu počítačových sítí velmi významnou roli, jelikož jejím hlavním úkolem je starat se o směrování a adresování k čemuž využívá patřičné protokoly. Tím nejvýznamnější je protokol IP neboli Internet Protocol. Internet Protocol je dnes velmi rozmanitý a existuje celkem ve dvou verzích, a to IPv4 a IPv6.

1.5.1 Adresace multicastu v sítích IPv4

Multicast je často nazýván jako skupinové vysílání, z čímž se pojí jeden velmi důležitý prvek a tím jsou tzv. skupinové adresy. Multicastové adresy pomáhají jednoznačně identifikovat skupinu uživatelů, kteří vyžadují poskytované data a usnadňují tak komunikaci s danou skupinou příjemců. Celosvětová organizace IANA, která dohlíží na přidělování IP adres, jednoznačně stanovuje pro technologii IP Multicast adresní prostor třídy D. Adresní třída D je jednoznačně specifikována hodnotami prvních čtyř bitů 1110 prvního oktetu. V desítkové soustavě tato sekvence bitů dává hodnotu 224 čímž lze jednoznačně určit, zda se jedná o multicastovou adresu. IPv4 adresa se ovšem skládá celkem ze čtyř oktetů po osmi bitech (32 bitů) což umožňuje vytvářet multicastové adresy v rozsahu 224.0.0.0 až 239.255.255.255. [2]



Obrázek 1.5: *Adresace multicastu v IPv4 [17]*

1.5.2 Registrace příjemců do multicastové skupiny

Multicastové vysílání se zakládá na skupinových adresách, které označují skupiny uživatelů přijímající distribuovaná data. Nicméně pro příjemce multicastu je nutné, aby se do skupin dokázali zaregistrovat. Pro tento účel je vytvořen jednoduchý mechanizmus v podobě protokolu, který dokáže informovat síť o snaze uživatele připojit se k dané skupině. Protokol je nazýván jako IGMP (Internet Group Management Protocol), který je využíván pouze v sítích založených na IPv4 adresách. Pro sítě IPv6 tento protokol již nelze využít. [2]

1.5.3 Přesměrování multicastu

V přesměrování skupinového vysílání je důležitá především rychlost. Pokud uživatel sleduje např. živě streamovaná videa nebo konference, potřebuje, aby data dostával co možná nejrychleji a ideálně v reálném čase. Pro tento účel se používá protokol UDP (User Datagram Protocol) pracující na Transportní vrstvě referenčního modelu ISO/OSI. V porovnání s protokolem TCP, který se snaží, aby data byly doručeny v pořádku a celistvá je protokol UDP přímým opakem. UDP nezáleží na tom, aby data dorazily v pořádku, ale aby byly doručeny co nejrychleji i za cenu ztráty paketu. [2]

1.5.4 Internet Group Management Protocol (IGMP)

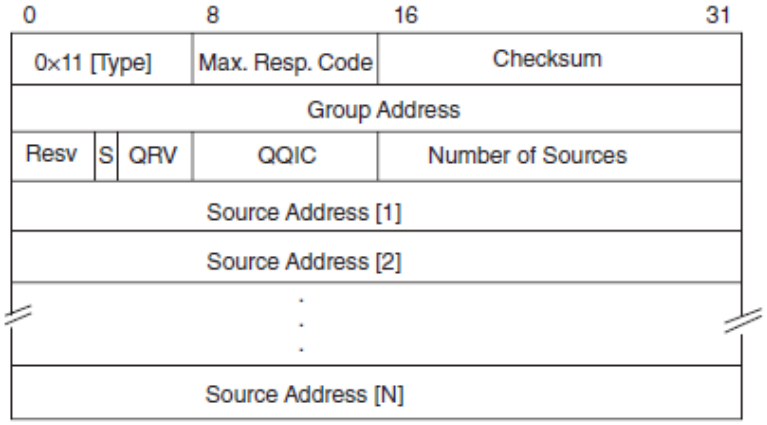
Registrace uživatelů do multicastové skupiny je z pohledu celé komunikace klíčové a pro zajištění její dynamičnosti je využíván protokol IGMP v sítích IPv4. Protokol IGMP umožňuje právě výměnu potřebných informací o změnách ve skupině, např. o přihlašování či odhlašování uživatelů. Komunikace probíhá s využitím zpráv mezi uživateli a směrovačem, který se nachází nejblíže dané skupiny a v pravidelných intervalech se doptává, zda je její součástí alespoň jeden aktivní uživatel. V první verzi protokolu jsou implementovány dva druhy zpráv. Jedná o zprávu typu MQ a MR. Zprávy typu MR se využívají k oznámení vstupu uživatelů, kteří se chtějí připojit do skupiny, a tato zpráva je zaslána k nejbližšímu směrovači. Směrovači je dán impuls k přijetí uživatele do skupiny, kterému musí směrovat multicastové data. Pro zajištění aktuálnosti přihlášených uživatelů je využívána zpráva druhého typu MQ. Směrovač se touto zprávou v pravidelných intervalech ptá zařízení, jestli jsou i nadále aktivní a chtějí přijímat distribuovaná data. Pokud zařízení nedokáží odpovědět na tři po sobě jdoucí zprávy tohoto typu, směrovač ukončí směrování dat. [2]

Podobně jako spousta jiných protokolů, tak i IGMP procházel postupem času vývojem. Právě díky vývoji je rozšířen o dvě verze IGMPv2 a IGMPv3, kde každá z nich dosahuje vylepšení oproti předchozím.

IGMPv2 lze považovat za následníka první verze tohoto protokolu. Podobně jako první verze i zde jsou obsaženy zprávy typu MQ a MR, které plní stejnou funkci. Co lze ovšem považovat za nadstavbu oproti první verze je přidání nového typu zprávy pro zvýšení efektivnosti komunikace mezi směrovačem i uživatelem. Zpráva je označována jako LG neboli Leave Group. Pomocí zprávy LG mohou uživatelé multicastové skupiny jednoznačně sdělit směrovači svůj záměr o opuštění skupiny. Díky tomu lze snížit vytížení přenosové trasy, a to zejména v sítích, které tvoří mnoho uživatelů. Pokud směrovač zprávu příjme, je nucen odeslat zprávu MQ. Je to z důvodu toho, že pokud uživatel, který zprávu LG poslal, byl zároveň posledním v dané skupině, nemá směrovač důvod pokračovat v distribuci dat. [2]

Třetí a poslední verze protokolu IGMP je ze všech nejsofistikovanější. Oproti předchozím dvěma verzím, jako jediná umožňuje filtrovat zdroje. Filtrováním lze určit, respektive zakázat více než jeden zdroj multicastového vysílání a dává uživateli možnost výběru konkrétních zdrojů. K tomu slouží dva rozdílné módy INCLUDE a EXCLUDE. INCLUDE je mód jehož užitím lze směrovači sdělit členství uživatele v určité skupině a seznam zdrojů, které si přeje přijímat.

Protipólem je mód EXCLUDE, který má stejnou funkčnost ovšem s rozdílným pojetím seznamu zdrojů. Zatímco u módu INCLUDE je vytvořen seznam za účelem povolení zdrojů, u módu EXCLUDE je seznam využit k jejich zakázání. [2]



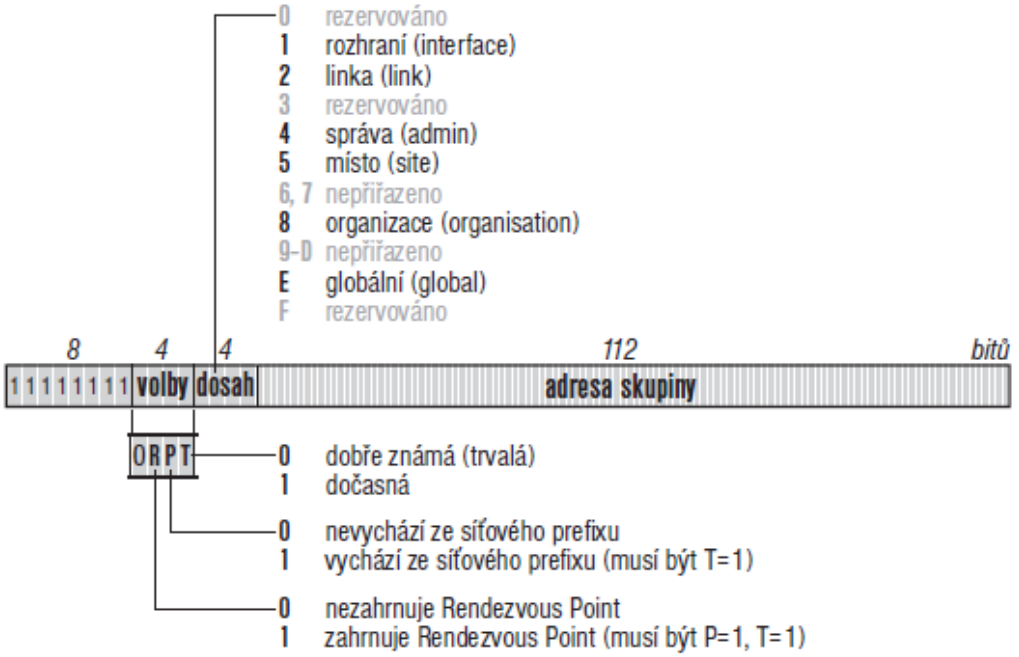
Obrázek 1.6: *IGMPv3 zpráva [2]*

1.6 Multicast v IPv6

Pro skupinové vysílání multicastových dat lze využít nejen sítě, které podporují protokol IPv4, ale také sítě zakládající se na adresách IPv6. Stejně jako IPv4 sítě tak i zde se multicastová komunikace skládá z příjemců a zdrojů dat, kteří vyžadují, respektive poskytují multicastová data. V porovnání se sítěmi v IPv4 je princip multicastového přenosu totožný, rozdílnost zde ovšem je. Mezi dvěma základními rozdíly, které se v sítích IPv6 nacházejí, jsou rozdílné adresování a protokoly pro vyhledávání a registraci uživatelů do skupiny.

1.6.1 Adresace multicastu v sítích IPv6

Podobně jako v IPv4 sítích tak i zde je směrování multicastového provozu založeno na skupinových adresách, které pomáhají k identifikaci skupin uživatelů vyžadující směrovaná data. Adresy v sítích IPv6 se skládají celkem z 128 bitů, což poskytuje více možností, jak s těmito adresami pracovat oproti 32 bitovým IPv4 adresám. Co se týče struktury multicastových adres, tak opět lze adresy jednoznačně identifikovat pomocí prvních 8 bitů zleva. Tyto bity jsou nastaveny na hodnotu 1 což po převodu do hexadecimální soustavy stanovuje hodnotu bitů jako ff. Za těmito bity následuje další sekvence čtyř bitů, která je označována jako tzv. volby. Sekvence těchto bitů není ovšem využita celá, jelikož první bit má nulovou hodnotu a nemůže být využit k specifickému účelu. Zbylé 3 bity mají jednoznačné označení R (rendezvous point), P (prefix) a T (transient). Bit s označením T dává najevo, zda je identifikátor skupiny přidělen trvale nebo pouze dočasně. Trvale přidělené adresy spravuje společnost IANA, nicméně dočasné adresy si mohou aplikace vytvářet dle potřeby. V pořadí třetí sekvence, která se opětovně skládá ze 4 bitů je tzv. dosah. Dosah nebo taky dosah skupiny stanovuje, jak daleko mohou jednotliví uživatelé od sebe být. Nastavení dosahu je velmi rozmanité a umožňuje nastavení celkem šestnácti hodnot z nichž pouze deseti je přidělen určitý význam. Poslední součást je pole 112 bitů sloužících pro adresu skupiny. [1][2]

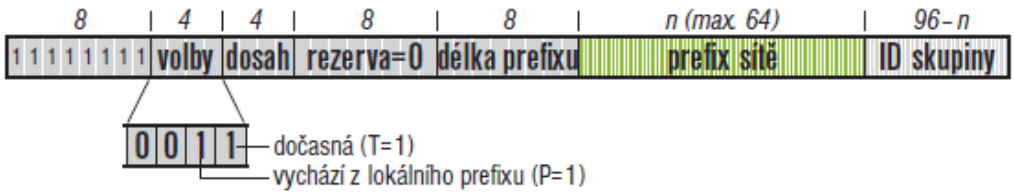


Obrázek 1.7: *Multicastová adresa v IPv6 [1]*

Pro zajištění dostupnosti streamovaných multicastových dat je v prácí používán dosah označený jako E, tedy globální. Vzhledem k přenosu skrze několika subnetů a požadované dostupnosti v celé topologii sítě je nastavení globálního dosahu dostačující pro distribuci dat. S dosahem je možné manipulovat dle potřeby a požadavků na dostupnost.

1.6.2 Adresy obsahující příznak P - Adresy vycházející z prefixu sítě

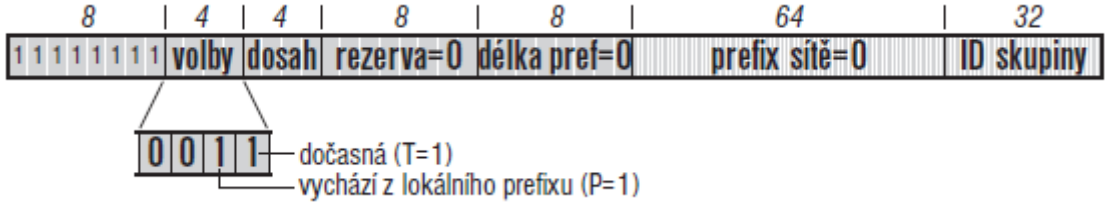
V multicastové komunikaci existují tzv. skupinové adresy vycházející z individuálních neboli adresy založené na prefixu sítě. Tyto adresy byly zavedeny z potřeby ulehčit vytváření jednoznačných adres, aniž by bylo nutné zjišťovat jejich existenci. Jednoduché vytvoření se zakládá na použitém prefixu sítě ve struktuře adresy. Jelikož je tento prefix celosvětově jednoznačný, stačí zajistit jeho jednoznačnost pouze v rámci sítě, čímž je jednoznačnost zajištěna celkově. S adresami vycházející z prefixu sítě úzce souvisí parametr P obsaženého v sekvenci 4 bitů, která je označena jako volby. Pokud parametr obsahuje hodnotu 1, oznamuje, že identifikátor skupiny byl vytvořen tímto způsobem. S parametrem P je v určité souvislosti spojen také parametr T. Pokud je hodnota parametru P nastavena na 1 musí být tato adresa dočasná, což lze zajistit právě pomocí parametru T. [1]



Obrázek 1.8: *Struktura multicastové adresy vycházející z prefixu sítě [1]*

1.6.3 Adresy obsahující příznak P - Adresy pro Source Specific Multicast

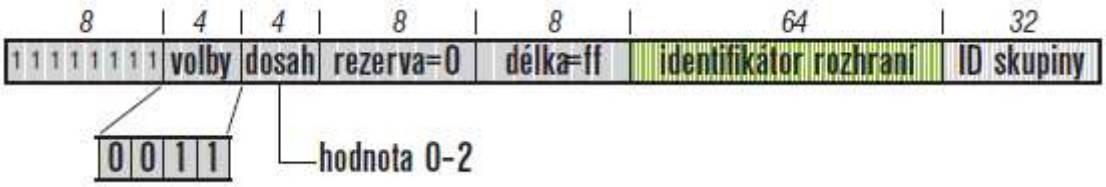
Další skupinou adres jejichž součástí je parametr P jsou adresy spadající do oblasti Source Specific Multicast. Tyto adresy mají velice jednoduchou strukturu a používají se pouze v případě přenosu dat z jednoho zdroje skupině příjemců, kteří jsou součástí multicastové skupiny. V porovnání se strukturou skupinové adresy vycházející z prefixu sítě nemusí tyto adresy obsahovat délku prefixu a samotný prefix sítě. Vlivem nastavení nulových hodnot těchto polí se struktura celé adresy podstatně zkracuje a prefix adres spadajících do SSM je ff3x::/96. [1]



Obrázek 1.9: *Struktura multicastové adresy pro SSM [1]*

1.6.4 Adresy obsahující příznak P - Adresy vycházející z rozhraní

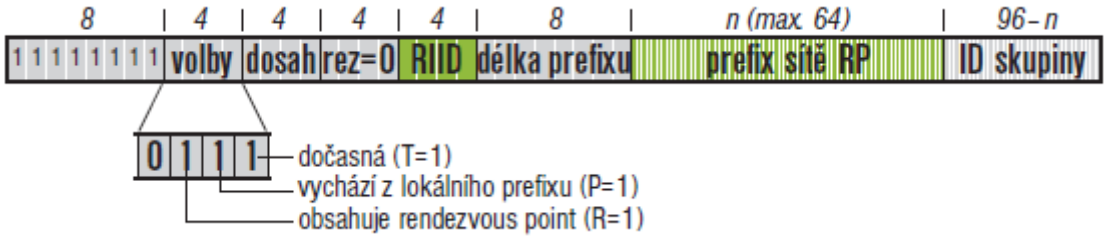
Adresy vycházející z rozhraní představují určitou konkurenci pro adresy založené na prefixu sítě. Mají ovšem jedno podstatné omezení, a tím je jejich dosah, který nemůže být větší než jedna linka. Výhodou adres je, že zařízení je mohou vytvářet sami, aniž by docházelo k totožnému vytváření mezi sousedy právě díky identifikátoru rozhraní. Identifikátor rozhraní musí být vždy jednoznačný. Struktura adres je velmi podobná adresám založených na prefixu sítě. Čím se ovšem liší, je hodnota délky prefixu nastaveného na ff (11111111) a místo identifikátoru sítě se uvádí identifikátor rozhraní. [1]



Obrázek 1.10: *Struktura multicastové adresy vycházející z rozhraní [1]*

1.6.5 Adresy obsahující příznak R - Adresy obsahující Rendezvous Point

Asi nejsofistikovanější typem adres, které se v multicastové komunikaci využívají, jsou tzv. adresy obsahující Rendezvous Point. Tento typ adres je úzce spjat s protokolem PIM (Protocol Independent Multicast) v režimu SM (Sparse Mode). Součástí sítě, která pracuje právě v tomto režimu, musí obsahovat RP neboli shromaždiště dat. Adresa RP je ve struktuře adres zakomponovaná a rozdělena do dvou částí. První je prefix sítě, který se nachází opět za délkou prefixu a jeho struktura se oproti předchozím případům nemění. Druhou částí je ovšem identifikátor rozhraní neboli RIID, který je vložen místo posledních čtyř bitů z rezervovaného pole. Jelikož tyto adresy mají všechny příznaky R, P a T hodnotu 1 začínají prefixem ff70::/12. [1]



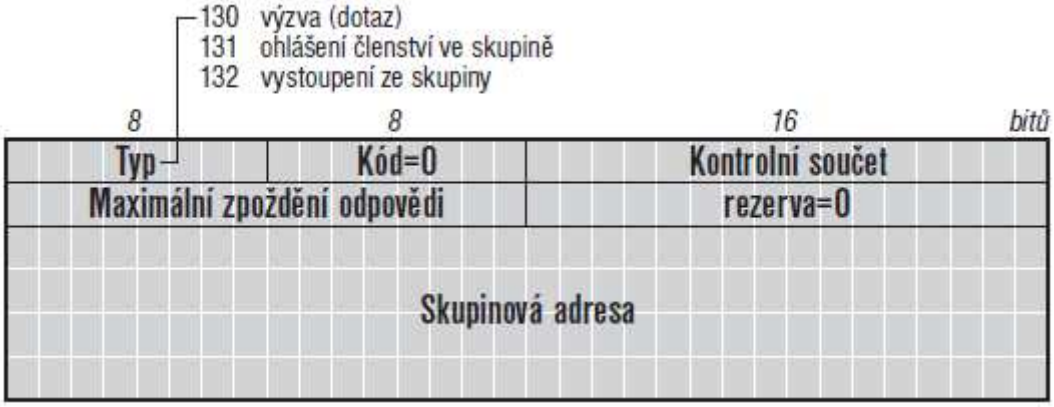
Obrázek 1.11: *Struktura multicastové adresy obsahující RP [1]*

1.6.6 Multicast Listener Discovery (MLD)

Směrování multicastového provozu není využíváno pouze v sítích založených na IPv4 adresování, ale také v sítích využívající protokol IPv6. Pro uživatele respektive příjemce tohoto provozu je vytvořen protokol MLD, jehož využití se vztahuje pouze k sítím založených na IPv6 adresování. Podobně jako u protokolu IGMP, který nelze využít v sítích IPv6 tak i MLD není možné využít v sítích IPv4. Základní informace pro směrovače určených ke směrování multicastu je zjištění příjemců ve skupinách, které obsahují alespoň jednoho uživatele. K tomu v IPv6 slouží protokol MLD. MLD se momentálně vyskytuje ve dvou verzích MLDv1 a MLDv2, jejichž základní principy jsou podobné protokolům IGMPv2 a IGMPv3. [1][2]

MLDv1 z anglického Multicast Listener Discovery Version 1 je prvním protokolem z rodiny MLD, sloužící pro zjištění přítomnosti a registraci příjemců do multicastové skupiny. Zatímco v druhé verzi tohoto protokolu lze vytvářet seznamy zdrojů, které zdroje povolují případně zakazují v jejich příjmu, je zde funkčnost omezenější a jedná se o protipól protokolu IGMPv2. Obdobně jako u protokolu používaného v sítích IPv4 i zde se využívají tří typy zpráv, díky kterým probíhá komunikace mezi směrovači a skupinami příjemců. Zprávy nesou číselné označení 130, 131 a 132, kde prvně zmíněná je definovaná jako výzva a slouží především směrovačům. Zbylé dvě zprávy využívají skupiny ke sdělení úmyslu o opuštění či vstupu do skupiny. Ke vstupu do skupiny je vyčleněná zpráva 131. Jestliže uživatel vstoupí do skupiny, je zpráva tohoto typu odeslána na její adresu a směrovači je dán impulz k přidání skupiny do seznamu vysílaných za předpokladu, že skupina na seznamu ještě není. Pokud v tomto seznamu je, jde o pouhé přidání uživatele vyžadujícího multicast. Příjemci multicastu ovšem mohou z multicastových skupin odcházet a k tomu slouží zpráva 132. Jedná se o zprávu, pomocí které uživatel sděluje svůj úmysl opustit skupinu a nepřijímat tak data. Zpráva se ovšem neposílá na adresu skupiny, ale na adresu ff02::2, která zahrnuje veškeré směrovače na dané lince. S touto zprávou souvisí také povinnost směrovače zjistit, zda má skupina alespoň jednoho uživatele, který multicast stále vyžaduje. K tomu lze využít prvně zmíněnou zprávu 130, která je definovaná jako výzva respektive dotaz. Dotazy využívají především směrovače pro zajištění aktuálního počtu sledujících ve skupinách a tím i povinnost směrovače zanechat skupinu v seznamu vysílaných. Zpráva je odeslána na adresu zjišťované skupiny což může způsobit hromadné odeslání a množení odpovědí od jejich uživatelů. Aby k této situaci nedocházelo, musí jednotlivý uživatelé navzájem spolupracovat a využít tzv. časovač. Časovač je nastaven každému zařízení neboli uživateli na náhodnou hodnotu jejíž maximální velikost je omezena velikostí pole Maximální zpoždění odpovědi. Po vypršení časovače, posílá dané zařízení zprávu o své přítomnosti na adresu skupiny.

Příjmem této zprávy vzniká povinnost pro zbylé uživatele zrušit své časovače a zabránit tak hromadění odpovědí. Veškerá námaha spojená se zprávou 130 závisí také na skutečnosti, který uživatel si vynucuje odchod ze skupiny. Pokud se odhlašuje právě ten uživatel, který se naposled do skupiny přihlašoval není směrovači jasné, jestli je ve skupině ještě někdo, kdo by multicast přijímal. Z tohoto důvodu musí směrovač podniknout nezbytné kroky týkající se zprávy 130 pro zjištění aktuálnosti skupiny. Výjma klasického dotazování směrovače existuje i tzv. obecný dotaz. Obecný dotaz je využit pro případy, kdy uživatelé neohlásí odchod ze skupiny, a právě z tohoto důvodu směrovače musí pokládat tyto obecné dotazy, jejichž skupinová adresa je nulová a jsou odesílány na adresu ff02::1. Adresa zahrnuje veškeré uzly na dané lince. [1][2]



Obrázek 1.12: *Formát zprávy MLDv1 [1]*

Následníkem a v pořadí druhý člen rodiny protokolů MLD je protokol MLDv2 neboli Multicast Listener Discovery Version 2. Druhá verze má prakticky identickou funkcionalitu jako IGMPv3 v IPv4 sítích. V porovnání s předchozí verzí MLDv1 je mnohem sofistikovanější a má jednu podstatnou výhodu. Výhodou je možnost výběru zdrojů multicastových dat, které lze povolit a případně i zakázat v jejich příjmu. Pro jejich omezování a povolení slouží dva režimy. Prvním je tzv. režim INCLUDE(L), kde L představuje souhrn všech zdrojů, které jsou povoleny pro příjem a uživatelé je smí využívat. Opakem je režim nazvaný jako režim EXCLUDE(L), kde naopak seznam L zdroje multicastu nepovoluje. S těmito režimy lze velice dobře pracovat a také je navzájem kombinovat, jelikož požadavky od různých uživatelů se mohou lišit. Pro tyto případy jsou stanoveny pravidla, které jednoznačně definují způsob, jak jednotlivé režimy kombinovat. [1][2]

U protokolu MLDv1 jsou zavedeny tři typy zpráv, z nichž dvě slouží výhradně pro příjemce multicastu. Pomocí těchto zpráv uživatelé informují o jejich záměru vstoupit, případně opustit skupinu. V případě MLDv2 tomu již tak není a obě situace jsou nahrazeny jedinou událostí. Tato událost je definovaná jako změna v příjmu skupin. Změna může obsahovat zahájení či ukončení příjmu skupin nebo také měnící se počet zdrojů, které jsou povoleny, respektive zakázány. V případě jakékoliv této změny je zapotřebí odeslat zprávu typu 143, označovanou jako Hlášení neboli Report na adresu ff02::16. Tato adresa zahrnuje veškeré směrovače na příslušné lince využívající protokol MLDv2. Pokud dochází ke změnám, může podobně jako v MLDv1 nastat situace, kdy směrovače nemají dobrý přehled o příjmu multicastu. Z tohoto důvodu je zde

zachován MLD dotaz, jehož funkce v podstatě kopíruje zprávu typu 130 rozšířenou o možnost přidat adresy zdrojů. Formy dotazu se mohou velmi lišit a existují celkem tři. První je nejjednodušší forma, která obsahuje nulovou adresu skupiny s dotazem na všechny příjemce multicastu. Druhou možností je dotaz, jehož cílem nejsou všechna zařízení, ale konkrétní skupina, kterou definuje pomocí její adresy. Poslední a zároveň nejrozšířenější je dotaz obohacený nejen o adresu skupiny, ale také o adresy zdrojů. [1][2]

1.7 Protocol Independent Multicast (PIM)

V multicastovém přenosu distribuovaného skrze IPv6 či IPv4 sítě, je zapotřebí dvou podstatných kroků. Jednak zjištění příjemců a jejich registrace do multicastových skupin pomocí protokolů rodiny MLD či IGMP, tak i distribuce dat ze zdroje skrze síť k příjemci. K tomu je využíván protokol označovaný jako PIM neboli Protocol Independent Multicast, jehož využití není omezeno v rámci typu adresování sítě. Protokol PIM, respektive protokoly zajišťují nejdůležitější, ale také nejtěžší část komunikace mezi zdrojem a příjemci. Mezi jejich základní vlastnosti patří řízení multicastových dat, zjištění rozložení příjemců v síti a vytvoření co možná nejefektivnější trasy, kterou by data byly doručeny. Nejedná se ovšem o jediný protokol, který lze pro tyto účely využít. Existuje i tzv. DVMPR (Distance Vector Multicast Routing Protocol) protokol, který ovšem není příliš vhodné využívat pro rutinní spojení. Vzhledem k této neefektivitě je dávána přednost rodině PIM složené ze čtyř protokolů: [1]

- PIM-Dense Mode (PIM-DM)
- PIM-Sparse Mode (PIM-SM)
- Bidirectional PIM (BIDIR-PIM)
- PIM Source Specific Multicast (PIM-SSM) [1]

1.7.1 PIM-Dense Mode

PIM-Dense Mode je základním protokolem, jehož princip distribuce dat je založen na předpokladu velkého množství příjemců, kteří chtějí přijímat multicastová data a mohou se nacházet téměř kdekoliv. Pokud data přicházejí na daný směrovač pracující v režimu Dense Mode, nakládá s nimi tak, že jednotlivá data rozesílá do všech svých rozhraní vyjma toho, ze kterého data dorazila. Tento druh distribuce může ovšem způsobit příchod dat z nečekaných směrů a v nejhorším případě dochází k vytvoření smyčky v síti, čímž data mohou kolovat nekonečně. Pro zamezení těchto nedostatků je využita RPF kontrola (Reverse Path Forwarding check). RPF pracuje s příchozími daty tak, že porovná jejich cestu k odesílateli pomocí směrovací tabulky. Pokud cesta vede rozhraním, kterým data dorazila, ponechá je beze změny a rozešle do všech rozhraní. V opačném případě data neodesílá. [1]

1.7.2 PIM-Sparse Mode

PIM-Sparse Mode pracuje v porovnání s režimem Dense Mode odlišným způsobem. Zatímco Dense Mode distribuuje data na základě předpokladu velkého množství příjemců, Sparse Mode očekává, že výskyt příjemců není příliš velký. Ke své činnosti využívá několika

mechanismů, čímž dosahuje vyšší efektivity v distribuci dat a stává se tak nejvyužívanějším v multicastové komunikaci. [1]

Klíčovým prvkem celé komunikace pomocí Sparse Mode je tzv. RP (Rendezvous Point), také označován jako shromaždiště. Jedná se o speciální směrovač, který představuje prostředníka mezi zdrojem multicastu a příjemci. Veškerá komunikace skupiny je tedy směřována pouze k tomuto směrovači a nikoli přímo ke zdroji. Je možné, aby všechny skupiny měli vlastní RP ovšem v reálné síťové struktuře bývá těchto shromaždišť pouze omezené množství. [1]

Důležitým pojmem v této komunikaci je sdílený strom neboli shared tree složený z kořene a větví. Kořenem tohoto stromu je RP a větve představují cestu k jednotlivým směrovačům, které mají zaregistrované příjemce vyžadující multicast. Celá komunikace probíhá tím způsobem, že zdroj vysílá data směrem k RP a poté jednosměrným sdíleným stromem jsou data distribuována dále až k příjemci. [1]

1.7.3 **Bidirectional PIM**

Bidirectional PIM protokol je nadstavbou protokolu PIM-SM, jehož odlišností je schopnost vytvářet obousměrné distribuční stromy. Jelikož režim Sparse Mode může vytvářet pouze jednosměrné sdílené stromy a nové stromy nejkratších cest, může docházet k velké zátěži směrovačů. Z tohoto důvodu se u protokolu Bidirectional PIM využívají obousměrné sdílené stromy, díky kterým mohou směrovače odesílat data, které se rozšíří k ostatním větvím. Tato distribuce má ovšem podstatnou nevýhodu v její nižší efektivitě směrování dat, jelikož cesta po které data kolují nemusí být vždy ideální. [1]

1.7.4 **PIM Source Specific Multicast**

Posledním protokolem z rodiny PIM je Source Specific Multicast, který lze považovat za částečný protokol v režimu Sparse Mode využívající pouze stromy nejkratších cest a je určen výhradně pro komunikaci s jediným zdrojem. Stromy nejkratších cest jsou označeny jako (S, G), kde symbol S definuje konkrétní zdroj multicastových dat a G skupinu příjemců. Vzhledem k jejich interpretaci rozlišují nejen adresu skupiny, ale také adresu zdroje, čímž jsou vhodné k přímé komunikaci se zdrojem. [1]

V porovnání protokolu PIM-SSM s Bidirectional PIM nebo PIM-Sparse Mode nepotřebuje tento protokol žádný rendezvous point, jelikož komunikace a žádost o připojení do distribučního stromu jsou posílány na adresu zdroje. [1]

2 Kodeky

Kodek je pojem převzatý z anglického slova codec a jedná se o souhrnné označení pro zařízení či programy, které slouží ke kódování či dekódování datového toku nebo signálu. Datový tok nebo taky stream a signály jsou díky těmto softwarovým nástrojům a zařízením transformovány do formátu, s kterým dokáže pracovat právě zvolený kodek. Všeobecně kodek nebo jeho název se skládá ze spojení dvou pojmů, a to kodér a dekodér z čehož vyplývá, že kodek musí zvládnout jak kódování dat, tak i jejich zpětné dekódování. [5][18][19]

Kodeky by měli splňovat dvě základní kritéria:

- Funkčnost oběma směry (kódování a dekódování),
- konkrétní implementace určitého formátu. [5]

U steamování videí pomocí multicastu je možné definovat nejen cílovou adresu pro příjemce, ale také způsob kódování video a audio dat. Tímto lze definovat základní dělení kodeků, které jsou vytvořeny pro video a audio data. [19]

2.1 Video kodeky

Video kodeky slouží ke zpracování video dat, kterým díky kompresních metod upravují jejich původní velikost. Velmi často je u video kodeků přítomna nejen video informace, ale také informace zvuková. Podobně jako rozdělení na audio a video kodeky i zde jsou video kodeky rozděleny do dvou základních skupin. Toto rozdělení závisí na ztrátovosti při kódování dat, a tak se rozdělují video kodeky na ztrátové a bezztrátové. [5]

2.1.1 Ztrátové video kodeky

U ztrátových video kodeků je podstatný jeden důležitý problém a to takový, že pokud jsou tyto kodeky využity, dochází ke ztrátě méně důležitých informací z původních dat. I když jsou informace považovány za méně důležité, ztráta těchto dat znemožňuje z nově vytvořených dat jejich zpětnou rekonstrukci. [5]

Mezi ztrátové kodeky lze zařadit kodeky z rodiny nesoucí označení DivX a XviD. Prvně zmíněná společnost DivX využívá toto označení pro své produkty, které vytváří a zahrnuje například video kodek MPEG-4. DivX je nástroj, který lze využít nejen pro kompresi videa, ale také pro jejich ořezávání a mnoho dalších úprav. Využívají například více proudové komprimování, jehož princip je založen na přechodech. V počátečním přechodu jsou získávány parametry a vlastnosti videa, které jsou zapsány do logu a v dalších přechodech komprimuje video dle informaci z prvního přechodu. Na rozdíl od Xvid, který obsahuje pouze samotný video kodek, DivX obsahuje i multimediální přehrávač a multimediální kontejner. [5]

Xvid je software, který je open source, tedy volně dostupný a může docházet k jeho modifikaci. Řadí se mezi ztrátové kodeky, a tudíž dochází ke ztrátě informací, čímž dosahuje velmi kvalitní komprese. [5]

2.1.2 Bezztrátové video kodeky

Dalším odvětvím, jsou video kodeky, které jsou bezztrátové. Jedná se o přesný opak kodeků, které se řadí mezi ztrátové, jelikož tyto kodeky umožňují zpětnou rekonstrukci původních dat z nově vytvořených. Bezztrátové video kodeky je vhodné využívat v případě, kdy je důležité, aby komprimovaná data po dekomprimaci byla stejná s daty původními. Jedná se například o text, kde je přítomnost, respektive nepřítomnost některých informací vhodná. [5]

Mezi bezztrátové druhy se řadí tzv. Huffyuv kodek, který je velmi rychlý a v dobrý podmínkách dosahuje i čtyřicetiprocentní komprimace. V prvopočátku měl být tento kodek využit pouze pro Microsoft Windows, ale nakonec byl zakomponován ve formátu FFmpeg, využívající např. VLC Media Player. [5]

Tabulka 2.1: Druhy video kodeků [19]

Název	Popis	Vlastnosti
mp1v	MPEG-1 Video	Kódování s přenosovou rychlostí až 1,5 Mbit/s
mp2v	MPEG-2 Video	Využíván pro DVD
mp4v	MPEG-4 Video	Vhodný pro streamování, IPTV a konference
WMV1	Windows Media Video 7	Video kodek vyvinutý společností Microsoft
WMV2	Windows Media Video 8	Nástupce kodeku WMV1
WMV3	Windows Media Video 9	Nejnovější verze kodeku WMV
H264	H.264	Vhodný pro přenos s vysokým rozlišením
H265	High Efficiency Video Coding	Nástupce kodeku H.264 s podporou UHD

2.2 Audio kodeky

Při streamování multimediálního obsahu lze definovat i formát zvukové informace, tedy použitý audio kodek. Audio kodeky oproti video kodekům neslouží ke zpracování video dat, ale pouze pro kódování či dekódování informací týkající se zvuku. Audio kodeky se stejně jako u videa dělí na dvě základní skupiny, které závisí na ztrátovosti při kódování dat, čímž jsou stejně jako video kodeky rozděleny na ztrátové a bezztrátové. [5]

2.2.1 Ztrátové audio kodeky

Podobně jako u video dat i zde dochází ke ztrátě informací ze zdrojových dat či signálu. Nejedná se ovšem o data, které by poslechu výrazným způsobem uškodily, ale převážně o informace, které lidský sluch nevnímá, případně si je člověk neuvědomuje. K těmto kodekům se řadí např. kodek MP3, WMA (Windows Media Audio) či AAC (Advanced Audio Coding). [5]

2.2.2 Bezztrátové audio kodeky

Bezztrátové audio kodeky umožňují zpětnou rekonstrukci audio dat z nového komprimovaného souboru, jelikož nezpůsobují ztrátu zvukových informací a jsou využity především v hudebním segmentu. Díky tomu, že nedochází ke ztrátě informací je kvalita dat prakticky nezměněná, a to i při snížení velikosti souboru. [5]

Tabulka 2.2: Druhy audio kodeků [19]

Název	Popis	Vlastnosti
mpga	MPEG audio	Komprese audio dat pro MP1, MP2
mp3	MPEG Audio Layer 3	Datový tok až 320 kbit/s s maximální kompresí 95 %
mp4a	MP4 audio	Vhodní pro IPTV, videokonference a streamování
WMA	Windows Media Audio	Možnost využití tzv. certifikovaného šifrování
AAC	Advanced Audio Coding	Až 48 kanálový zvuk
AC3	Audio Coding-3	Přenosová rychlost až 640 kbit/s
FLAC	Free Lossless Audio Codec	Bezztátový kodek s kompresí až 60 %

2.3 MPEG (Moving Picture Experts Group)

Kompresní formát MPEG nebo taky Moving Pictures Experts Group je oblíbený a mezi uživateli velmi rozšířený formát, který se skládá z několika standardů, které slouží ke kódování a dekódování audio i video dat či signálu. Společně se čtyřmi standardy tvoří velmi robustní rodinu kodeků využívající kompresní algoritmy, které byly vytvořeny skupinou stojící za vznikem tohoto formátu. Skupina vznikla roku 1988 a během pětiletého snažení roku 1993 dokončila první ze standardů MPEG-1. Celkem se skládá ze 4 standardů a to: [5][20]

- MPEG-1
- MPEG-2
- MPEG-3
- MPEG-4 [5]

2.3.1 MPEG-1 (Moving Picture Experts Group 1)

MPEG-1 je prvně vytvořeným kodekem, který skupina vydala roku 1993 a patří mezi ztrátové kodeky, čímž dochází ke ztrátám méně důležitých informací. Z jeho vlastností vyplývá schopnost kódování digitálních audio či video dat s přenosovou rychlostí v rozmezí 0,9 až 1,5 Mbit/s. Díky jeho kompatibilitě je využíván v mnoha technologiích a jeho téměř nejznámější součástí je standard pro audio data a tím je kodek MP3. Standard je složen celkem z pěti částí: [5][21]

- Systémy
- Video

- Zvuk
- Testování shody
- Referenční software [5]

2.3.2 MPEG-2 (Moving Pictures Experts Group 2)

Nástupcem kodeku MPEG-1 vytvořeného roku 1993 je kodek nesoucí označení MPEG-2, jehož výhodou oproti svému předchůdci je možnost pracovat s prokládanými snímky. Tento standard je určen pro generické kódování pohybujících se snímků a zvukových dat a jedná se ztrátový kodek s přenosovou rychlostí 1,5 až 15 Mbit/s. V porovnání s dalšími druhy kodeků, jako H.264 nebo H.265 není standard MPEG-2 schopen dosáhnout takové účinnosti, ovšem díky vysoké kompatibilitě s různými typy zařízení a aplikací je i nadále využíván. Jeho využití je soustředěné především na kompresi digitálních dat pro televizní vysílání nebo je využit pro standard známý jako DVD. Tento kodek nabízí nejen práci s prokládanými snímky, ale také možnost vyššího rozlišení, a tedy i vyšší kvality v porovnání s MPEG-1. [5][22]

2.3.3 MPEG-3 (Moving Pictures Experts Group 3)

Standard MPEG-3 byl v počátku vytvářen jako náhrada za kodek MPEG-2, který by měl využití především pro oblast televizního vysílání s požadovanou HD kvalitou neboli HDTV s dosažením přenosové rychlostí okolo 20 až 40 Mbit/s. Ovšem díky již dostatečným kvalitám jeho předchůdce, nebylo nutné v projektu nadále pokračovat a stal se součástí již vyvinutého druhého kodeku MPEG-2. [5][23]

2.3.4 MPEG-4 (Moving Pictures Experts Group 4)

Posledním standardem rodiny MPEG je čtvrtá verze MPEG-4 vyvinuta koncem roku 1998, která převzala mnoho funkčních vlastností od svých předchůdců MPEG-1 a MPEG-2, obohacenou o mnoho dalších funkčních prvků. Tyto prvky nabízí kupříkladu podporu pro 3D vykreslování za pomocí nástroje VRLM. Využití tohoto kodeku je poměrně široké a slouží například ke: [5][24]

- Streamování na internetu,
- IP televize a televizní vysílání,
- videokonference,
- multimediální přehrávače. [5][24]

Oproti svým předchůdcům nabízí standard MPEG-4 mnoho výhod, které vylepšují poskytování služeb zejména v oblasti televizního příjmu. Mezi výhody se řadí:

- Efektivnější forma kódování v porovnání s MPEG-2, což umožňuje v jednom multiplexu přenášet až 10 kanálů, místo 5,
- odolnost vůči chybám, čímž dosahuje vyšší robustnosti,
- snížení datového toku a zachování kvality obrazu (vhodné pro HDTV),
- možnost zakódovat data s různých médií, jako je audio, video nebo řeč,
- kódování placených televizních kanálů. [5]

2.4 H.264

Kodek H.264 je nástupcem formátu H.263 a jedná se o kompresní standard pro video data. Standard může být znám i pod označením MPEG-4 AVC (Advanced Video Coding) a jedná se o velmi rozšířený kodek, jehož využití se vztahuje především k digitálnímu příjmu televizního signálu. Lze jej ovšem využít i pro standard DVD nebo Blue-ray. [25][26]

Cílem projektu je vytvořit kodek, který je schopen poskytnout vysokou kvalitu obrazu i v případě nižších přenosových rychlostí v porovnání s kodeky jako je MPEG-2 nebo H.263. Dále tento kodek zajišťuje určitou flexibilitu pro jeho použitelnost v širokém spektru různých aplikací a systémů: [25][26]

- Videa s vysokým rozlišením
- televizní vysílání,
- telefonní systém,
- paketové sítě,
- uložiště DVD či Blue-ray. [26]

Formát H.264 je vyvinut ve spolupráci dvou skupin VCEG (Video Coding Experts Group) a MPEG jejichž vzájemná kooperace na tomto projektu nese společné označení JVT (Joint Video Team). [26]

2.5 H.265

Kodek H.264 je v současnosti hojně využíván pro digitální příjem televizního signálu DVB-T, avšak postupem času bude nahrazen novějším a lépe propracovaným formátem H.265. Formát H.265 neboli HEVC (High Efficiency Video Coding) existuje v několika verzích a je jedním z nejnovějších druhů kodeků vyvinutý v roce 2013. Tento kodek je využíván pro nový standard digitálního pozemního televizního vysílání DVB-T2, který má zcela nahradit současný standard DVB-T. [4][27][28][29]

Nový kodek H.265 nabízí řadu výhod v porovnání se staršími formáty, mezi které lze zařadit:

- Dvojnásobná efektivita komprese v porovnání s kodekem H.264, což přispívá ke zvýšení kvality obrazu,
- podpora UHD technologie a 8K,
- větší velikost makrobloků (64x64). [27]

Vyjma kvalitnějšího obrazu a mnoha dalších výhod je s kodekem spojena i řada nevýhod, které mohou způsobovat řadu problémů a zde se řadí například:

- Náročné kódování,
- vysoké nároky na výpočetní výkon (až desetkrát náročnější). Jedná se o jeden z největších důvodů, který přispívá k vyšší využitelnosti kodeku H.264. [27]

2.6 FFmpeg

Formát FFmpeg lze považovat za multimediální systém nebo taky aplikaci, která se řadí mezi tzv. open source nástroje. Open source nástroje jsou volně dostupné a lze je libovolně používat ke zpracování ať už audio či video obsahu. U formátu FFmpeg je ovšem využitelnost mnohem rozsáhlejší a tím nabízí uživateli širší škálu možností, jak s tímto formátem pracovat. Mezi základní funkční prvky patří nejen práce s audio a video daty, ale také možnost streamování a filtrování. FFmpeg je znám především v operačních systémech Linux, pro které je primárně určen a je využíván např. programem VLC Media Player. Využití tohoto formátu ovšem není omezeno pouze na operační systém Linux a lze jej využít i pro systémy jako jsou Microsoft Windows, Mac OS či BSD. Asi nejdůležitější součástí standardu ze souboru knihoven je knihovna libavcodec. Tato knihovna je zodpovědná za komprimaci a případnou dekomprimaci audio a video dat. [30][31]

2.7 FLAC

Formát FLAC neboli Free Lossless Audio Codec je bezztrátový kodek, který podobně jako FFmpeg patří k softwaru, označovaného jako open source neboli softwaru, který je volně dostupný. Kodek FLAC je mnohdy spojován s kodekem MP3 díky jejich podobnosti. Podobnost mezi těmito formáty je ale velice diskutabilní, jelikož je mezi nimi jeden podstatný rozdíl. Zatím co formát FLAC se řadí k bezztrátovým audio kodekům, čímž nedochází ke ztrátě informací z původních dat, formát MP3 je kodek ztrátový a zde ke ztrátě dat dochází. Tyto ztracená data již nelze zpětně získat. Z tohoto důvodu je dobré rozlišit situaci, kdy je vhodnější použít MP3 nebo FLAC. Pokud je zapotřebí, aby veškeré informace z audio souboru zůstaly zachovány, je výhodné využít právě kodek FLAC. I když je tento kodek zařazen mezi bezztrátové kodeky, dosahuje velice dobrých výsledků v komprimaci audio souborů a je schopen zredukovat soubor až na polovinu z původní velikosti. [32][33][34]

2.8 Vorbis

Kodek Vorbis je jeden z dalších formátů, který se řadí mezi open source nástroje, využívající ztrátovou kompresi a je využíván pro komprimování a dekomprimování audio dat. Co se týče jeho použitelnosti pak lze tento kodek velice často spatřit ve spojení s kontejnerem typu Ogg a tento název je často zakomponován i jako součást názvu tohoto kodeku neboli Ogg Vorbis. Kodek byl vytvářen od roku 1993, ovšem k prvnímu vydání použitelné verze došlo až v roce 2002. Účinnost a kvalita kodeku v porovnání s jinými formáty je na velmi dobré úrovni a dosahuje velice dobrého kompresního poměru se stejnou případně i vyšší kvalitou a je konkurenceschopný například s kodekem MPEG-4. [35][36][37]

2.9 AAC

Advanced Audio Coding je kodek využívaný pro audio data a k jejich komprimaci či dekomprimaci využívá ztrátovou kompresní metodu. Kodek AAC využívá velice dobré kódování a jeho kvalitativní prvky výrazným způsobem převyšují i formát MP3, jehož kvalitu překonal

nejen tento kodek ale i kodek Vorbis. S vývojem kodeku AAC se začalo již ve dvacátém století a je patentován skupinou MPEG. Mezi uživateli ovšem není příliš známý a ani používaný i přes jeho vysoké kvality. Pokud jde o strukturu kodeku, pak se jedná formát vytvořený kombinací dvou kodeků MP3 a AC3 s mnoha výhodami: [5][38]

- Podpora vícekanálového zvuku a to až 48 kanálů,
- nižší výkon pro dekomprimaci,
- jednoduchost a všestrannost,
- vyšší rozsah kmitočtů,
- vyšší efektivita komprese. [5]

2.10 AC3

AC3 z anglického názvu Audio Coding 3 je kodek vyvinutý společností Sony, určený především pro zpracování audio dat v oblasti filmů. Tento kodek se řadí mezi jedny z nejlepších formátů a byl vyvinut především pro soubory obsahující prostorový zvukový záznam, který lze využít například u formátu DVD či Blu-ray. Mezi základní vlastnosti kodeku patří: [5][39][40]

- Bitová hloubka 24 bitů,
- kmitočty od 48 do 96 kHz,
- prostorový zvuk až 7.1,
- přenosová rychlost až 640 kbit/s. [5]

2.11 MP3

Asi nejznámějším a nejrozšířenějším audio kodekem na světě je kodek MP3, někdy označovaný jako MPEG-1 nebo MPEG-2 Audio Layer 3. Jedná se o ztrátový typ kodeku, odstraňující části zvuku, které jsou pro lidský sluch nepostřehnutelné. Pomocí MP3 lze dosáhnout poměrně značně vysoké komprimace a tím i snížení velikosti původních souborů i na pouhou jednu desetinu velikosti. Základním datovým tokem kodeku je 128 kbit/s, tento parametr je možné s ohledem na kvalitu upravit a tím dosáhnout i vyšší kvality. Mezi vlastnosti lze zařadit: [5][41][42][43]

- Datový tok 128 až 320 kbit/s,
- komprimace až 95 %,
- podpora formátu až 5.1. [5]

2.12 Apple Lossless

Apple Lossless kodek, jak již vyplývá z názvu je vytvořen společností Apple a jedná se o bezztrátový kodek sloužící pro komprimaci a dekomprimaci audio dat určených především pro uživatele, kteří využívají zařízení této společnosti. Často bývá označován pomocí dvou základních zkratek, a to ALAC (Apple Lossless Audio Codec) nebo ALE (Apple Lossless Enconder). Vývoj formátu Apple Lossless začal již ve dvacátém století a jeho vznik se datuje k

roku 2004 a následně od roku 2011 je kodek zařazen k open source nástrojům, čímž se stal volně dostupným. Základní charakteristikami jsou: [5][44][45]

- Až 8 kanálů zvuku,
- bitová hloubka až 32 bitů,
- vzorkovací frekvence až 384 kHz,
- komprimace až na 60 % původní velikosti souboru,
- nízká náročnost na dekomprimaci. [5][45]

2.13 WMA

Od společnosti Microsoft, která je známá především svými operačními systémy, je možné využít také kodeky vyvinuté pro potřeby jejich uživatelů. Windows Media Audio je základním kodekem, vyvinutý pro zpracování audio souborů, za jehož vznikem stojí výzkumník Henrique Malvare. V porovnání s konkurenčními kodeky bývá nejčastěji spojován s kodekem MP3, jehož popularitu mezi uživateli výrazným způsobem ovlivňuje. WMA disponuje dvěma základními výhodami, z nichž první je možnost komprimace zvukových záznamů vyšší rychlostí. Navýšením rychlosti komprimace napomáhá zvýšení kvality zvuku za cenu nižších požadavků na paměť. Další podstatnou výhodou je tzv. certifikované šifrování. Certifikované šifrování slouží především pro společnosti a vydavatele z hudebního odvětví, jelikož tento typ šifrování se zaobírá ochranou práv autorů. Díky tomu lze zabránit případnému padělání a další nelegální distribuce mezi uživateli. Ovšem ve struktuře kodeku se nachází i řada nevýhod. Podstatným neduhem kodeku je, že v mnoha případech může dojít k nedokonalému zaznamenání vyšších frekvencí, při nižších přenosových rychlostech a tím i k poklesu kvality, což pro náročnější uživatele může působit nepříznivě. [5][46]

2.14 WMV

Podobně jako u kodeku Windows Media Audio, vyvinutého společností Microsoft, tak i kodek Windows Media Video patří mezi základní články společnosti. Nejedná se ovšem o kodek sloužící pro komprimaci a dekomprimaci audio dat, ale pouze pro data jejichž informace slouží pro video. Kodek WMV dokáže pracovat se soubory, jejichž velikost dosahuje podstatně vyšších hodnot a tyto soubory zmenší se zachováním stejné kvality. Díky tomu je kodek využitelný především v oblasti streamování videí. Vzhledem k tomu, že se jedná především o kodek, vyvinutý společností Microsoft je složité najít přehrávač, který by byl schopen kodek přehrát. S tímto problémem se lze setkat především u operačních systémů Linux a Mac OS. Postupem času ovšem vývojáři dokázali formát WMV zakomponovat i do jiných systémů a je možné se s tímto kodekem setkat i v systémech, jako je právě Linux využívající VLC Media Player. [47]

3 Linuxové programy, jejich vlastnosti a konfigurace

Jednou z hlavní součástí práce, je možnost výběru linuxových programů, díky kterým lze vytvořit streamovací server, příjemce streamovaných dat a mnohem více rozmanitějších konfigurací. Konfiguracemi je možné nastavit např. rozhraní, směrovací protokoly či režim multicastového přenosu. Pro zajištění veškeré funkcionality, ať už z pohledu streamování nebo vytváření směrovačů v topologii sítě, je zvoleno pět základních programů pomoci kterých, lze provést konfigurace k zátěžovému testování v laboratoři. Jedním z programů, je program VLC Media Player. Díky tomuto nástroji lze zajistit vše z pohledu streamování, ať už se jedná o stream server či příjemce multicastu. Dále je využit na jednotlivých počítačových zařízeních program LXD. Program LXD je jeden z virtualizačních nástrojů, sloužící pro vytváření virtuálních strojů na daném zařízení s možností výběru různých distribucí Ubuntu. Zdaleka nejdůležitějšími programy, jsou programy Xorp, MRD6 a SMCRoute. Pomocí těchto nástrojů, lze vytvářet z jednotlivých počítačů směrovače, zastávající funkci běžných směrovačů, které jsou známy např. od firmy Cisco.

3.1 Xorp

Program Xorp je základním stavebním kamenem pro vytvoření potřebné topologie a nástrojem k zátěžovému testování. Jedná se o program, využívaný jako routovací platforma, která je libovolně rozšířitelná a řadí se mezi volně dostupné open source nástroje. Díky libovolnému rozšíření, lze u této platformy zavádět např. nové funkční prvky, protokoly a mnohem více. Podstatnou výhodou je možnost pracovat jak v IPv4, tak v IPv6 a jedná se o jednu z mála open source platforem, nabízející práci s multicastovým přenosem dat a práci s multicastem vůbec. IP protokoly ovšem nejsou jediné, které platforma Xorp podporuje. Vyjma těchto protokolů lze využít i řadu již zavedených protokolů jako jsou OSPF, RIP, BGP, PIM či MLD, které jsou důležité zejména pro přenos multicastu. Co se týče podpory operačních systémů platformy, pak není omezena pouze v rámci Linuxu, ale je zde také možnost využití v operačním systému Windows či systému BSD. [48][49]

Počátky vývoje programu Xorp sahají až do vzdálené Kalifornie v USA, kde celý program vznikal, jako projekt na ICSI Center for Open Networking neboli ICON. Cílem projektu bylo dosáhnout platformy, která by byla otevřená a dávala tak možnost všem uživatelům vyzkoušet si bezplatně různé konfigurace a práci s protokoly, za kterou jiné společnosti vyžadují finanční spoluúčast. Pokud dojde ovšem k porovnání služeb a podpory mezi společnostmi jako je Xorp nebo Cisco, má společnost Cisco významnou převahu. [49]

Xorp jako takový se skládá z mnoha procesů. Operátor neboli uživatel programu nemá přístup ke všem těmto procesům, ale pouze k procesům vybraným, které jsou bezprostředně nutné ke konfiguraci směrovače. Je to dámo tím, že společnost nechce umožnit přístup k procesům a dovolit tak znalost veškeré vnitřní struktury programu. Jedná se také o míru bezpečnosti nejen pro samotné uživatele, kteří by mohli nenávratně poškodit tuto vnitřní strukturu a opětovně by museli program instalovat, ale také pro bezpečnost programu samotné společnosti. Součástí

struktury procesů je jediný proces, který je určen pro správu samotného směrovače. Tímto procesem je tzv. xorp_rtrmgr, který by měl být obsažen v adresáři /usr/local/xorp/sbin/. Význam adresáře je navíc umocněn tím, že zde musí být umístěn konfigurační soubor, který je potřeba spustit. Pokud by se konfigurační soubor nacházel kdekoliv jinde, nedojde k jeho spuštění. V neposlední řadě je potřeba spouštět konfiguraci pod právy superuživatele root, přepínačem mínus b a daný konfigurační soubor by měl obsahovat koncovku boot případně koncovku conf. [50]

Součástí adresáře /usr/local/xorp/sbin/ je kromě procesu pro správu směrovače i proces xorpsh sloužící k přístupu do shellu samotného programu Xorp. Xorpsh představuje informativní náhled a možnost výpisu informací, kterými lze ověřit korektnost konfigurace. [50]

```
xorp_rtrmgr -b soubor.boot
```

```
xorp_rtrmgr -b soubor.conf
```

```
xorpsh
```

Vývojáři aplikace Xorp poskytují uživatelům i řadu základních konfigurací, které mohou sloužit jako inspirace pro začínající správce. Konfigurace se nacházejí v adresáři /xorp.ct/xorp/rtrmgr/config/ a jsou uživatelům volně k dispozici. Pokud je potřeba tyto konfigurační soubory použít je zapotřebí jejich strukturu pozměnit dle vlastní potřeby. [50]

3.2 Konfigurace Xorp

Aby bylo možné konfigurovat vlastní soubory a využívat předpřipravené protokoly je nutné program stáhnout, naistalovat, zkompilovat a připravit systém k jeho prvnímu spuštění. Společnost Xorp dává uživatelům možnost stáhnout program na oficiálních stránkách společnosti, kde nabízí různé možnosti přístupu. Stažení lze provést např. pomocí webové služby GitHub případně prostřednictvím Live CD. [50]

Pokud je program Xorp v počítačovém zařízení obsažen je možné jej nainstalovat a zkompilovat. K tomu, aby bylo možné program zkompilovat, je nutné získat sadu potřebných balíčků, které definuje samotná společnost. Instalace je provedena pomocí příkazu apt-get install za předpokladu, že má uživatel oprávnění superuživatele root. V opačném případě je nutné před tento příkaz doplnit sudo. [50]

```
# apt-get install build-essential git scons libboost-all-dev
libssl-dev libncurses5-dev libpcap-dev traceroute flex bison
```

Úspěšnou instalací všech balíčků, je možné přistoupit k instalaci a kompilaci celého programu. Ke kompilaci je vyčleněn balíček scons. Pokud je tento balíček spuštěn je potřeba vytrvat, jelikož kompilace může trvat i delší dobu. Po dokončení kompilace lze Xorp naistalovat a připravit k jeho spuštění. Veškerá instalace a kompilace se provádí v adresáři xorp.ct/xorp. [50]

```
# scons
```

```
# scons install
```

V této fázi je již program nainstalován a zkompilován, ovšem samotný systém musí být připraven na jeho použití. Pokud uživatel určitým způsobem změní konfiguraci směrovače, systém musí být schopen tuto změnu realizovat. K tomu je nutné v systému vytvořit skupinu xorp a přidat do této skupiny superuživatele root. [50]

```
sudo groupadd xorp
sudo usermod –a –G xorp root
```

Po dokončení veškeré instalace a přípravy systému na první spuštění programu je možné vytvářet vlastní konfigurační soubory. V konfiguračních souborech musí být definovány veškeré potřebné parametry, které jsou podmíněné funkčností tohoto souboru i celého programu Xorp. Mezi základní parametry, které je nutné specifikovat, patří např. rozhraní, protokoly a mnoho dalších funkcí, které jsou k dispozici v uživatelském manuálu. [50]

3.2.1 Konfigurace rozhraní

```
interfaces {     … počátek oddílu interfaces
restore-original-config-on-shutdown: bool  … obnovení
konfigurace při náhlém vypojení rozhraní (defaultně false)
     interface text {      … definice rozhraní např. eth0..
          description: text     … vlastní popis rozhraní
          mac: macaddr          … definice MAC adresy rozhraní
          mtu: uint             … definice maximální hodnoty MTU
          disable: bool         … aktivace/deaktivace rozhraní
          discard: bool         … nastavení vyřazení rozhraní
          unreachable: bool     … nastavení dostupnosti rozhraní
          management: bool      … nastavení rozhraní pro správu
          vif text {            … nastavení vif rozhraní
               disable: bool    … aktivace/deaktivace vif
               address IPv4-addr/IPv6-addr {    … IP adresa
               prefix-length: int(1..32)/int(1..128) … prefix
               destination: IPv4-addr/IPv6-addr… cílová adresa
               disable: bool    … aktivace/deaktivace adresy
               }     … ukončení oddílu pro IP adresu
          }     … ukončení oddílu pro vif rozhraní
     }     … ukončení oddílu pro definici rozhraní
}     … ukončení oddílu interfaces [7]
```

Základem každé konfigurace směrovače je nastavení jednotlivých rozhraní, skrze které směrovače komunikují s připojenými zařízeními, a to i v opačném sledu. Výše uvedená konfigurace je demonstrací všech možných nastavení rozhraní na daném směrovači. Veškeré informace, týkající se všech rozhraní, musí být uvedeny v oddílu nazývající se interfaces, ohraničeny složenými závorkami. Uvnitř tohoto oddílu lze uvádět veškeré informace definované vývojáři a samotnou konfiguraci všech aktuálně připojených rozhraní, skrze které je směrovač schopen komunikovat. [7][50]

3.3 LXD

Program LXD je jedním z dalších programů, využívaný v této práci pro realizaci streamování dat pomocí multicastu. Jedná se o virtualizační open source nástroj, označovaný jako správce Linuxových kontejnerů, které vytváří z již předem zavedených obrazů různých systémů. Nejedná se tedy o typický virtualizační nástroj jako je např. program VirtualBox, který vytváří virtuální stroje, ale o program, který vytváří a využívá Linuxové kontejnery. Kontejnery jsou koncipovány tak, aby mohly nabídnout prostředí, které by bylo co možná nejpodobnější prostředí, získaného pomocí virtuálních strojů, ovšem bez nutností simulování hardwaru či dalších akcí spojených s procesem virtualizace. Jedná se o tzv. kontejnerovou virtualizaci. Tento způsob virtualizace nabízí možnost běhu jediného operačního systému, nad kterým lze vytvářet různá prostředí neboli kontejnery, jejichž funkčnost není vzájemně závislá. Díky tomuto způsobu virtualizace není potřeba instalovat na jednotlivé kontejnery celý operační systém, čímž lze ušetřit systémové prostředky a poskytnout řadu výhod: [6][51][52][53]

- Nižší náročnost na systémové zdroje.
- Nižší náročnost na techniku.
- Velká škálovatelnost.
- Vyšší bezpečnost.
- Možnost sdílení hardwaru.
- Lepší kontrola prostředků.
- Libovolný přechod mezi jednotlivými kontejnery.
- Možnost využití různých distribucí Linux v jednotlivých kontejnerech. [53][54][55]

Způsob kontejnerové virtualizace ovšem není bezchybný a je možné zde narazit na řadu nevýhod, které mohou mít nepříznivý vliv jako jsou:

- Nejedná se o pravou virtualizaci.
- Možné problémy s AppArmor či SELinux. [53]

Jednou z velmi důležitých součástí kontejnerové virtualizace je nástroj zvaný hypervisor. Pokud dochází k vytváření kontejnerů, pak každé prostředí musí mít nadefinovanou skupinu virtuální zařízení. Složení těchto zařízení může být velmi proměnlivé, ale musí obsahovat všechny zařízení, které jsou nutné pro funkčnost jednotlivých operačních systémů nainstalovaných ve virtuálních kontejnerech. Tuto úlohu splňuje právě hypervisor, díky kterému dochází k jejich alokaci na fyzickém stroji. [6]

Mezi virtualizační nástroje využívající virtuální kontejnery a všeobecně kontejnerovou virtualizaci nepatří pouze program LXD. Mezi další nástroje lze zařadit např. nástroj LXC nebo LXCFS. Prvně zmíněná technologie LXC je opětovně jako LXD volně dostupným open source nástrojem sloužící pro vytváření virtuálních Linuxových kontejnerů. LXD je ovšem v porovnání s LXC mnohem sofistikovanější a poskytuje řadu lepších a také novějších funkčních prvků sloužící pro správu kontejnerů. V případě LXCFS se jedná o souborový systém sloužící pro zkoumání nedostatků jádra operačního systému Linux. [51][56][57]

3.4 Konfigurace LXD

Konfigurace programu LXD je velice prostá a nevyžaduje žádné doplňující balíčky, v porovnání s programem Xorp, jelikož vše je součástí jediného balíčku. Balíček programu nese shodné označení s názvem programu a je možné jej naistalovat pomocí příkazu apt install. Předtím než je program naistalován je vhodné provést tzv. update a upgrade v terminálu systému Linux. Díky tomu je zajištěna aktuálnost všech balíčku, které jsou k dispozici a které mohou uživatelé použít. [58]

```
# apt-get update
# apt-get upgrade
# apt install lxd
```

Po závěrečné instalaci je možné přikročit ke konfiguraci programu a provést jeho nastavení. K tomuto inicializačnímu procesu slouží příkaz init, po kterém je uživatel vyzván k zadání potřebných parametrů. Všechny parametry je možné nastavit dle vlastního uvážení, případně ponechat vše v základním nastavení, které má program zabudované. Pomocí parametrů lze nastavit např. režim shlukování neboli clustering, díky kterému jsou veškeré instance LXD schopné sdílet a využívat jedinou databázi. Dále pak program nabízí možnost nastavení uložiště, MAAS serveru, síťového bridge, používání IPv4 a IPv6 adres, přístup do sítě či automatický update. [58][59]

```
# lxd init
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no)
[default=yes]:
Would you like to connect to a MAAS server? (yes/no)
[default=no]:
Would you like to create a new local network bridge? (yes/no)
[default=yes]:
What IPv4 address should be used? (CIDR subnet notation, "auto"
or "none") [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, "auto"
or "none") [default=auto]:
```

```
Would you like LXD to be available over the network? (yes/no)
[default=no]:

Would you like stale cached images to be updated automatically?
(yes/no) [default=yes]:

Would you like a YAML "lxd init" pressed to be printed? (yes/no)
[default=no]:
```

Dokončením instalace a konfigurace je možné vytvářet potřebné kontejnery s operačními systémy, kterými program disponuje. LXD má k dispozici řadu distribucí operačního systému Linux v podobě obrazů tzv. images, které využívá pro vytvoření kontejnerů. Seznam obrazů je možné vyhledat pomocí příkazu image list, u kterého je možné definovat, o jakou distribuci se jedná, např. ubuntu či debian. Vyhledáním názvu potřebného obrazu je klíčové, jelikož tento název je součástí při vytváření kontejnerů pomocí příkazu launch. [58]

```
# lxc image list images:

# lxc image list images: | grep –i ubuntu

# lxc launch images:{distro}/{version}/{arch} {container-name}
[58]
```

Vytvořené kontejnery pomocí příkazu launch nejsou ovšem nastaveny tak, že by docházelo k automatickému přesměrování do těchto prostředí. K tomu, aby bylo možné s kontejnery pracovat, je potřeba se k virtuálním prostředím přihlásit využitím příkazu shell. Díky tomu je možné přistupovat k jednotlivým prostředím a pracovat s nimi dle potřeby. Pro opuštění kontejneru slouží příkaz exit. [58]

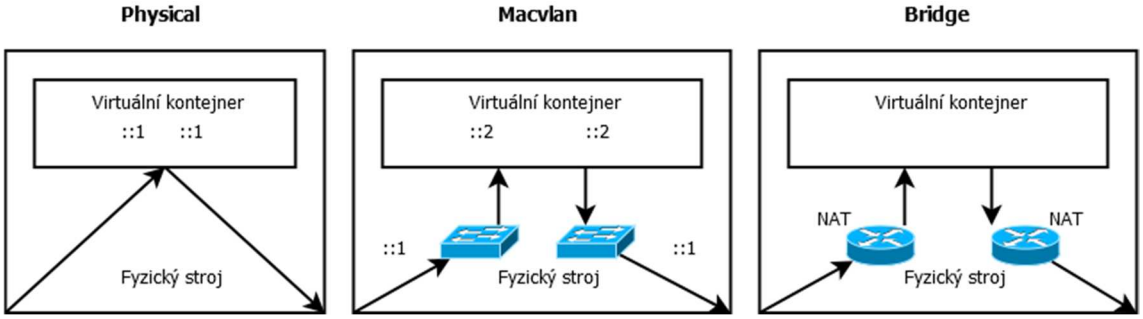
Příkazy jako jsou shell nebo exit nejsou jediné, které slouží k práci s virtuálními kontejnery. Pro tyto účely jsou využívány např. příkazy pro aktivatci (start), deaktivaci (stop), restartu (restart) či odstranění (delete) kontejnerů. [58]

```
# lxc shell containerName

# lxc start containerName

# lxc stop containerName

# lxc restart containerName

# lxc detete containerName [58]
```

Pro vytvoření směrovačů ve virtuálních kontejnerech je zapotřebí do virtuálního prostředí přidat množství rozhraní, skrze které by mohlo směrovací zařízení komunikovat. Jelikož zařízení obsahují i více jak dvě nebo tři rozhraní, lze v případě programu LXD a virtuálních kontejnerů přidat podstatně více rozhraní než u běžných směrovačů.

```
# lxc config device add containerName interfaceName nic name=
interfaceName nictype=physical/macvlan/bridge
parent=interfaceName [69]
```

Rozšíření počtu rozhraní v kontejneru LXD lze provést pomocí základního příkazu config device add, čímž je dán sytému impulz k přidání rozhraní do kontejneru. Systém ovšem vyžaduje zavedení parametrů, podle kterých lze rozhraní přidat. Prvním parametrem je název kontejneru, který je cílem rozhraní neboli containerName. Tímto parametrem je nutné specifikovat určitý kontejner, aby systém byl uvědomen, který kontejner je jeho cílem. Dále je potřeba specifikovat název rozhraní a jeho typ. Název rozhraní a typ LXD používá pro potřebnou práci s rozhraním. Co se týče typu rozhraní, jedná se o rozhraní sloužící pro směrovače a nejvhodnějším je NIC neboli síťové rozhraní (network interface). Následuje položka name u které je nutné specifikovat opět název rozhraní. Nejedná se ovšem o název, který slouží pro práci programu LXD, ale o název, pod kterým bude rozhraní vystupovat uvnitř kontejneru. Název rozhraní lze libovolně změnit. V konečné fázi je potřeba nastavit tzv. parent a nictype. U hodnoty parent jde opět o název rozhraní, ale ne o rozhraní, které má souvislost s kontejnerem či LXD, ale o název rozhraní připojeného přímo k fyzickému stroji. Název tohoto rozhraní jde zjistit pomocí příkazu ifconfig nebo ip a v terminálu. Co se týče parametru nictype, pak se jedná o nejdůležitější volbu, kterou lze učinit. Parametr nictype nabízí celkem tři možnosti, jak rozhraní nastavit a to physical, macvlan nebo bridge. Každý z těchto režimů pracuje odlišným způsobem, ovšem pro potřebu směrovače je nejvhodnější režim physical, ať už z pohledu bezpečnosti či manipulace se zařízením. Režim physical pracuje tak, že rozhraní připojené k fyzickému stroji vnoří přímo do kontejneru. Rozhraní je tímto způsobem viditelné pouze z daného kontejneru a nikoli na fyzickém stroji. [69]



Obrázek 3.1: Režimy pro nictype

3.5 VLC Media Player

Multimediální programy jsou nezbytnou součástí každého systému a jde o velmi užitečné nástroje pro uživatele, kteří pracují s audio či video daty. Většina společností vyvíjející své operační systémy, často vytváří i vlastní multimediální nástroje pro multimediální obsah a snaží se jejich funkčnost co nejvíce přizpůsobit uživateli. Typickým příkladem společnosti pracující v této oblasti je společnost Microsoft. Společnost Microsoft společně s operačním systémem Windows, vyvinula i program Windows Media Player, který slouží převážně pro uživatele systému této společnosti. Nejedná se ovšem o jediný nástroj sloužící pro práci s audiovizuálními daty.

Rozmanitost programů sloužící k práci s audiovizuálními daty je ve skutečnosti příliš rozsáhlá a existuje nemalý počet softwaru, který lze k tomuto účelu využít. Jedním z programů,

který ovšem lze řadit mezi nejvyužívanější a tím i nejznámější je program VLC Media Player. Nástroj VLC Media Player je využit i v této práci jako zdroj dat neboli stream server sloužící k distribuci audio a video dat, a také pro jednotlivé příjemce vyžadující jejich příjem. Vznik programu VLC se datuje k roku 1996, jako akademický projekt studentů, který postupem času prochází vývojem a dnes je spravován organizací známou jako VideoLAN. V současnosti je vyvinuta nejnovější verze 3.0.4 Vetinari pro operační systémy Linux, Windows a Mac OS. S postupem času lze ovšem předpokládat příbytek novějších a inovativnějších verzí, a to i pro další operační systémy, jako jsou Chrome nebo Android. Podobně jako na začátku vývoje i dnes se jedná o volně dostupný open source nástroj, jehož přítomnost lze očekávat v kterémkoli systému. Volba programu VLC je velice vhodná zejména z pohledu streamování. Nejedná se pouze o obyčejný multimediální přehrávač, ale také o nástroj, který je možné využít jako stream server. Díky tomu lze pohodlně vytvořit zdroje multicastu šířícího skrze počítačovou síť. [60][61][62]

```
vlc –vvv video.mp4 --ttl 12 --loop --sout
'#transcode{vcodec=codecName,acodec=codecName,vb=,ab=}:std{acces
s=,mux=,dst=[multicastAddress]:1234}'      … stream videa z PC

vlc –vvv v4l2:///dev/video0 --ttl 12 --sout
'#transcode{vcodec=codecName,acodec=codecName,vb=,ab=}:std{acces
s=,mux=,dst=[multicastAddress]:1234}'      … stream pomocí
kamery
```

Pomocí výše uvedené konfigurace neboli kódu, lze vytvořit potřebný stream server, jehož účelem je distribuce multicastových dat. Kód je navržen tak, aby zdrojem multicastu mohly být předem připravené audiovizuální data, případně video kamera s možností různých nastavení. Nastavit lze např. hodnota ttl, smyčka loop nebo forma dat odchozího streamu. VLC Media Player vyjma spuštění serveru pomocí terminálu s tímto kódem, nabízí možnost ovládání pomocí grafické nadstavby s využitím klasických oken. [63][65]

Součástí kódu jsou dvě podstatné části, jejichž nastavení ovlivní budoucí formu dat odchozího streamu. Jedná se o tzv. modul transcode a standard neboli std. Prvně zmíněný modul transcode slouží k definici audio a video kodeků společně s přenosovými rychlostmi. Co se týče přenosové rychlosti u jednotlivých formátů je důležité ověřit, jak je daný kodek schopen data nastavit, jelikož maximální hranice přenosových rychlostí může být odlišná. Podpora kodeků u programu VLC ať už pro audio nebo video data je velice rozsáhlá, ale mezi zdaleka nejvyužívanější kodeky patří: [63][65]

- H264 (video kodek)
- H265 (video kodek)
- mp4v (video kodek)
- mp4a (audio kodek)
- mp3 (audio kodek)
- flac (audio kodek) [64]

V druhé části u modulu standard jde především o způsob doručení nebo ukládaní dat. K tomu slouží tři parametry access, mux a dst. Parametrem access lze určit výstup ze zdroje, který může být nastaven v pěti různých variantách: [65]

- udp: slouží pro streamování pomocí unicastu nebo multicastu
- http: slouží pro streamování pomocí HTTP
- https: slouží pro streamování pomocí HTTPS
- file: slouží pro ukládaní dat do souboru
- mmsh: slouží pro streamování pomocí MMS [65]

Následuje parametr zvaný mux. Parametr mux určuje tzv. metodu zapouzdření streamovaných dat neboli kontejner, který zajistí jejich synchronizaci a jedná se o povinný parametr, tudíž jeho nastavení je podmíněné funkčností celého streamu. Základním a nejvyužívanějším kontejnerem je kontejner ts, který je možné využít v kombinaci s libovolnou metodou access, ale lze nastavit i další kontejnery jako: [65]

- ps: lze využít pro access typu file nebo http
- ogg: lze využít pro access typu file nebo http
- avi: lze využít pro access typu file
- mpeg1: lze využít pro access typu file nebo http [65]

Posledním a velmi důležitým parametrem je cíl přenášených dat označen jako dst. Hodnota parametru dst závisí na nastavení metody access. [65]

- udp: uvádí cílovou unicast nebo multicast adresu
- http: uvádí cílovou adresu, port a cestu
- https: uvádí cílovou adresu, port a cestu
- file: uvádí cestu pro uložení souboru
- mmsh: uvádí cílovou adresu, port a cestu [65]

Pokud je funkčnost vytvořeného stream serveru správná, lze z pohledu příjemce streamovaná data přijímat. Aby se příjemci mohli přihlásit k danému streamu, lze využít opět program VLC Media Player a níže přiložený kód. [66]

```
vlc –vvv udp://@[multicastAddress]:1234
```

3.6 VLC 3.0.4 Vetinari

Jednou z nejnovějších verzí programu VLC Media Player, kterou disponují i nejnovější distribuce Ubuntu 18.04, je VLC 3.0.4. Vetinari. S verzí 3.0.4 je disponováno i v této práci a slouží pro vytvoření potřebného stream serveru jako zdroje multicastu a také pro multicastové příjemce neboli uživatele. S postupem času dochází v počítačových sítích ke změnám v technologiích, systémech a dalších odvětvích. Účelem vývoje je především rozšíření služeb a možností pro uživatele, kteří tyto softwary využívají a kladou tak mnohem vyšší požadavky, například na kvalitu, rychlost a další parametry. S každou novější verzí softwaru přichází i řada změn v jeho struktuře a k zaštítění chyb, které mohly působit nepříznivě ve verzích předchozích.

Podobným systémem se řídí i společnost VideoLAN a proto i ve verzi VLC 3.0.4 Vetinari došlo k řadě změn: [67][68]

- Podpora kodeku AV1,
- lepší dekódovací schopnost kodeku HEVC v systémech Windows,
- lepší práce s videi nízké kvality,
- lepší zabezpečení v systémech Windows. [67][68]

3.7 SMCRoute

Program Static Multicast Routing Daemon zkráceně SMCRoute je nástrojem, pro správu multicastových routovacích tabulek zabudovaných v jádře operačního systému Linux. Jeho podpora se nevztahuje pouze na tento operační systém, tudíž je možné program využít i v jiných systémech, jako FreeBSD a další. SMCRoute je součástí systému jako balíček dostupný ke stažení a následnému použití jako alternativu pro multicastové směrovače pracující na dynamickém principu. [71][72]

```
# apt install smcroute
```

Podpora u programu je velice příhodná, jelikož umožňuje práci s multicastem jak v IPv4 sítích, tak v sítích IPv6 s očekáváním konfiguračního souboru s koncovkou conf v adresáři /etc/smcroute.conf. Po instalaci programu je vhodné pomocí příkazu dpkg zjistit adresářovou strukturu souborů obsažených v balíčku a zkopírovat návrh základního konfiguračního souboru do požadovaného adresáře. Pokud se zkopírovaný soubor v adresáři nachází, je možné přikročit k nastavení vlastní konfigurace pomocí příkazu nano, či jiného nástroje, který slouží pro úpravu souborů. [71][72]

```
# dpkg -L smcroute          … zjištění adresářové struktury

# cp /cesta/ke/konfiguracnimu/souboru/smcroute.conf
/etc/smcroute.conf    … zkopírování konfiguračního souboru

# nano /etc/smcroute.conf  … úprava konfiguračního souboru
```

V konfiguračním souboru se nachází veškeré informace spojené se směrováním multicastu včetně základního návrhu pro multicastové směrování v síti IPv4 využívající zdrojový strom (S, G) a sdílený strom (*, G). Sdílený strom je omezen funkčností pouze pro IPv4 sítě a nikoliv pro sítě IPv6. Strukturu základních návrhů je potřeba změnit dle vlastní potřeby a zejména upravit vzhledem k adresování v síti.

```
mgroup from eth0 group 225.1.2.3

mroute from eth0 group 225.1.2.3 source 192.168.1.42 to eth1
eth2

mgroup from eth0 group 225.3.2.1

mroute from eth0 group 225.3.2.1 to eth1 eth2
```

3.8 MRD6

Jedním z dalších programů sloužících pro distribuci multicastových dat je program IPv6 Multicast Routing Daemon neboli MRD6. Jedná se o program nebo taky daemon, jehož zakladatel Hugo Santos stanovil funkční prvky pouze v rámci IPv6 sítí a dostupnost je omezena v rámci systémů Linux nebo BSD. Vzhledem k tomu, že je program určen výhradně pro IPv6 sítě, je i podpora protokolů omezená v rámci těchto sítí. Protokoly jsou dostupné v nejnovějších verzích a protokol PIM lze využít v mnoha režimech: [73][75]

- MLDv1 a MLDv2
- PIM-SM
- PIM-SSM
- Stadická volba RP
- Dynamická volba RP[73][75]

Podobným způsoben jako u programu SMCRoute je i program MRD6 volně dostupný z balíčkového systému a nevyžaduje žádný externí přístup. Instalaci balíčku lze provést pomocí příkazu apt install, čímž se do souborové struktury naimportují i návrhy konfiguračních souborů, které je možné upravit. Na rozdíl od programu SMCRoute neobsahuje balíček MRD6 jeden, ale hned tři konfigurační soubory, jejichž struktura je odlišná. Tento program očekává soubor v určitém adresáři /etc/ s koncovkou conf. [74]

```
# apt install mrd6          … instalace programu MRD6
# dpkg -L mrd6              … zjištění adresářové struktury
# cp /cesta/ke/konfiguracnimu/souboru/mrd.conf /etc/mrd6.conf
                        … zkopírování konfiguračního souboru
# nano /etc/mrd6.conf       … úprava konfiguračního souboru
```

4 Konfigurace linuxových zařízení pomocí bash skriptů

Programování je v dnešní době velice populární a nabízí mnoho možností a funkčních prvků nejen pro uživatele, ale také pro vývojáře a programátory. V oblasti programování, lze rozlišovat dva základní druhy jazyků, a to jazyky skriptovací a kompilační. Oba jazyky mají odlišné využití i struktury, ale vývojově starším je jazyk kompilační. Kompilační jazyk je postaven na určitém základu a tím je zdrojový kód. Zdrojový kód je nutné vytvořit a následně pomocí překladače přeložit do tzv. strojového kódu. Pokud překlad proběhne v pořádku je možné program spustit a případně dále zdokonalit jeho funkčnost. Mezi kompilační programovací jazyky lze zařadit například C, Fortran, Assembler a mnoho dalších programovacích jazyků jako: [76][77]

- BASIC
- Pascal
- LabVIEW [76]

U skriptovacích jazyků jde především o zvýšení efektivnosti a dosažení určité úrovně automatizace, díky čemu dochází zejména k úspoře času, rychlejšímu vývoji a pro uživatele jednoduššímu použití. Výběr mezi skriptovacími jazyky je velmi rozsáhlý, ale k základu patří jazyky, jako Bash, Perl či Python. Python se díky svému intuitivnímu a jednoduchému použití stal velmi oblíbeným a mocným nástrojem, který se prosadil například v oblasti webových stránek a jedná se spíše o novější druh skriptovacího jazyka. [77][78]

Naopak mezi již známější a starší programy patří například Perl. Tento jazyk vyzývavě přejímá řadu rysů a podobností z programovacího jazyku C. Výhodami, kterými kdysi patřil mezi nejvýznamnější prvky, spočívala především v míře možností zpracování textu. Nicméně dnes tyto prvky lze spatřit i v řadě dalších jazyků a tuto přednost již nelze brát jako výhodu. Co však z pohledu Perlu nelze přehlédnout je jeho schopnost pracovat se systémovými prostředky, čímž dosahuje velké úspornosti. Ovšem i tato úspora prostředků má nepříznivý vliv na kvalitu a tím i čitelnost. Všeobecně syntaxe tohoto skriptovacího jazyka je velmi roztržitá a lze jednoduše vytvářet chyby způsobující nežádoucí chování. [77]

V linuxovém prostředí je asi nejtypičtějším skriptovacím jazykem Bash. Skriptovací jazyk Bash je plnohodnotným nástrojem pro programování v oblasti Unixových systémů a společně s dalšími jazyky se navzájem doplňují ale nekonkurují. S ostatními druhy nelze tento jazyk příliš srovnávat, jelikož Bash není vhodné využívat na všechny úkony v oblasti programování, a tak není možné, aby si jiné skriptovací jazyky navzájem konkurovaly s tímto typem. Bash skriptování nebo taky Shellové skripty nejsou nikterak složitými či jinak komplikovanými úkony za předpokladu alespoň základní znalosti syntaxe v příkazovém řádku unixových systémů. Skripty obsahují totiž zápis jednotlivých příkazů, které by uživatel byl nucen postupně zadávat v příkazovém řádku a vytváří tak velkou časovou úsporu a automatizaci úloh

pro dosažení výsledku. Shellové skripty nejsou ojedinělou záležitostí, která se týká výhradně unixových systémů, ale využití skriptů je možné aplikovat i pro operační systémy Windows. [77]

4.1 Bash skriptování

Pro vytvoření Bash skriptu v Unixových systémech není zapotřebí speciální prostředí, jako je tomu u různých kompilačních jazyků např. C nebo Java. K tomu, aby uživatel dokázal vytvořit skript je možné využít jednoduchý editor pro zpracování textu ať už v textovém či grafickém prostředí. V grafickém prostředí je možné použít programy, jako jsou např. editor gedit nebo gvim a v textovém prostředí nano, vim či mcedit. Vytvořený soubor by měl obsahovat specifickou koncovku sh, ale tato koncovka není podmínkou. Systém dokáže z obsahu souboru rozpoznat, že se jedná o skript a podle toho bude se souborem zacházeno. Aby mohl být skript spustitelný, je nutné prvotní nastavení neboli hodnota prvního řádku skriptu musí obsahovat kód, který informuje systém o jeho vytvoření: [79][80]

```
#!/bin/bash
```

Tento kód informuje interpret jazyka Bash o tom, že se jedná o skript a podle toho musí být se souborem zacházeno. Díky tomu je základní struktura skriptu navržena a lze vytvářet kompaktnější a rozmanitější konfigurace. [79][80]

```
# nano skript.sh            … vytvoření souboru
# chmod +x skript.sh        … nastavení práva spuštění
# . skript.sh               … spuštění skriptu
```

Aby bylo možné vytvářet kompaktní skripty je vhodné znát nejen základní syntaxi unixových systémů, ale také vytváření proměnných, podmínek a cyklů. Proměnné nejsou záležitostí pouze kompilačních jazyků, ale také jazyků skriptovacích a díky proměnným je možné např. ukládat, vypisovat a přepisovat potřebné hodnoty. [79][80]

```
promenna=obsahPromenne            … uložení obsahu do promenna
promenna2=$obsahPromenne          … uložení obsahu promenna do
proměnné promenna2
echo Hodnota promenne je: $promenna   … výpis obsahu promenna
```

4.1.1 Podmínka if

Podmínka if je jedna ze základních podmínkových výrazů a řídící struktura, jejíž vyžití lze aplikovat i ve skriptovacích jazycích. Struktura podmínky může být v různých situacích navržena ve třech tvarech, ale základní tvar podmínky neboli jednoduchý if je definován následně: [80][81]

```
if [ podminka ] ; then            … začátek podmínky
          Prikazy                 … příkazy k vykonání
Fi                                … konec podmínky
```

4.1.2 Cyklus While a For

Kromě podmínkových výrazů lze ve skriptech využít i tzv. cykly. Cykly jsou navrženy především k vykonání opakované činnosti definované pomocí příkazů a lze je rozdělit na dva základní druhy. Prvním je tzv. cyklus while. U tohoto typu vzniká největší nebezpečí zacyklení, jelikož příkazy, které obsahuje, se provádí do doby, než je výstupní hodnota nulová (skončí úspěšně). [80][81]

```
while [ logicky_vyraz ]
do
     prikazy
done
```

Naproti cyklu while je typ zvaný jako cyklus for. Cyklus for je odlišný především znalosti počtu vykonaných kroků. Zatímco u cyklu while není možné počet kroků specifikovat, u tohoto typu to možné je a tím přispívá k menší míře nebezpečí zacyklení. [80][81]

```
for promenna in seznam
do
     prikazy
done
```

4.2 Skript pro Stream server a příjemce

Pro urychlení práce na streamovacím serveru a zařízení příjemců je vytvořen skript, jehož struktura je navržena tak, aby zařízení obsahovalo všechny potřebné programy pro práci s multicastovými daty a byla zajištěna konektivita s příslušným směrovačem. Níže uvedené kódy jsou pouhé výtažky jednotlivých částí, nicméně celková struktura skriptu je uvedena v přílohách.

```
echo Provadim update repozitaru a instalaci programu pro server.
sleep 3
apt-get update                          … update
apt install -y vlc tcpdump wireshark  … instalace programů
```

V první části skriptu je instalace potřebných programů pro server i příjemce a update repozitářů. Po vykonání těchto kroků je možné přistoupit k nastavení příslušné IPv4 nebo IPv6 adresy a výchozí brány. Pokud nastavení proběhlo bez problémů je možné ověřit konektivitu se směrovačem, ke kterému je server či příjemce připojen. K ověření je zvolen cyklus while za podpory podmínkových příkazů if pro stanovení konečného rozhodnutí o konektivitě. Na počátku cyklu je stanovena pomocná proměnná pokusy, která omezuje funkčnost cyklu na celkový počet deseti pokusů, čímž je zabezpečena nebezpečná situace případného zacyklení. Dále je stanoven příkaz ping, pomocí kterého je sledována odezva na příslušný směrovač a závěrem skriptu jsou podmínky pro ukončení cyklu a stanovení konečného rozhodnutí o konektivitě.

```
((pokusy = 10))                          … maximální počet pokusů
while [[ $pokusy -ne 0 ]] ; do           … počátek cyklu while
     ping -c 3 IPv4_adresa/IPv6_adresa … ping na IP adresu
     rc=$?
     if [[ $rc -eq 0 ]] ; then
     ((pokusy = 1))                      … opuštění cyklu while
fi
((pokusy = pokusy - 1))
done
if [[ $rc -eq 0 ]] ; then                … stanovení konektivity
     echo "Server/Prijemce je pripojen."
else
     echo "Server/Prijemce není pripojen"
fi
```

4.3 Skript pro směrovače Xorp

Pro vytvoření síťové topologie složené z několika směrovačů je vytvořen skript, který zabezpečí veškerou činnost správce spojenou se směrovači a umožňuje jejich rychlejší a lehčí ovladatelnost. Skript společně s konfiguračním souborem (soubor.boot) a programem Xorp (xorp.ct-master.zip) je přítomen ve virtuálním kontejneru vytvořeného za pomocí programu LXD a v tomto kontejneru je skript spuštěn.

Spuštěním skriptu dojde v první řadě k instalaci programů, které jsou nutné pro instalaci programu Xorp a případnou manipulaci s konfiguračním souborem. K tomuto účelu je zapotřebí program unzip a nano.

```
echo Instaluji nastroje pro spravu smerovace.
apt-get install -y unzip nano tcpdump
unzip /home/ubuntu/xorp.ct-master.zip -d /home/ubuntu
mv /home/ubuntu/xorp.ct-master /home/ubuntu/xorp.ct
```

V dalším kroku je potřeba instalace balíčků pro program Xorp, aby mohl být ve virtuálním kontejneru správně naistalován a zkompilován. Definici balíčků stanovuje samotná společnost na oficiálních stránkách.

```
echo Instaluji potrebne balicky.
apt-get install -y build-essential git scons libboost-all-dev
libssl-dev libncurses5-dev libpcap-dev traceroute flex bison
```

Po ukončení instalace balíčků dochází k instalaci a kompilaci samotného programu Xorp a ověření jejich správnosti. K určení korektnosti instalace i kompilace je nutné zjistit, zda existuje soubor xorp_rtrmgr v adresáři /usr/local/xorp/sbin/, který zodpovídá za správu směrovačů.

```
cd /home/ubuntu/xorp.ct/xorp
scons
scons install
if [ ! -e /usr/local/xorp/sbin/xorp_rtrmgr ]; then
     echo "Instalace neprobehla spravne."
else
     echo "Kompilace a instalace probehla spravne."
fi
```

V poslední fázi skriptu dochází k přesunu konfiguračního souboru (soubor.boot) do požadovaného adresáře, vytvoření skupiny xorp, přiřazení uživatele root do této skupiny a spuštění směrovače. Přesun konfiguračního souboru do adresáře /usr/local/xorp/sbin/ je velmi důležitou součástí při spouštění programu. Program Xorp je koncipován tak, že očekává konfigurační soubory výhradně v tomto adresáři a nikde jinde. Pokud by se soubor nacházel v jiné části adresářové struktury, program by nebyl schopen soubor spustit.

```
cp /home/ubuntu/soubor.boot /usr/local/xorp/sbin/
cd /usr/local/xorp/sbin/
groupadd xorp              … vytvoření skupiny xorp
usermod -a -G xorp root    … přidání uživatele root do skupiny
/usr/local/xorp/sbin/xorp_rtrmgr -b
/usr/local/xorp/sbin/soubor.boot      … spuštění směrovače
```

4.4 Skript pro směrovače SMCRoute

Pro zařízení, které pracují s programem SMCRoute je vytvořen skript, zajišťující jejich připravenost na směrování multicastových dat s využitím základního konfiguračního souboru. Oproti programu Xorp je nutné nastavit směrování a přesměrování multicastových dat, jelikož tyto možnosti nejsou součástí souboru.

```
apt install -y smcroute tcpdump wireshark
sysctl -w net.ipv6.conf.all.forwarding=1
cp /usr/share/doc/smcroute/examples/smcroute.conf
/etc/smcroute.conf
```

5 Zátěžové testování v laboratorních podmínkách

Pro testování v laboratorních podmínkách jsou vybrány tři programy jejichž funkčnost je testována na vlastně vytvořené síťové topologii a jedná se o program Xorp, MRD6 a SMCRoute. Vyjma nástroje Xorp jsou všechny programy testovány v systému Ubuntu 18.04, jelikož funkčnost tohoto programu je omezená pouze do verze Ubuntu 14.04. Je to dáno tím, že nejnovější verze systému prošla značnými změnami, a to včetně knihoven C++ které jsou nyní nekompatibilní s tímto programem v nejnovější verzi Xorp 1.8.6. Problém s verzemi systému je vyřešen díky programu LXD, pomocí kterého jsou vytvořeny virtuální kontejnery s vyhovujícím systémem.

5.1 Vytvoření LXD kontejneru pro směrovač R3 (Xorp)

Virtuální kontejnery zajišťující kompatibilitu programu Xorp a distribuce Ubuntu jsou vytvářeny nástrojem LXD. Po instalaci a inicializaci programu je možné vytvořit příkazem lxc launch kontejner s požadovaným systémem a přikročit k definici rozhraní, které mají být jeho součástí.

```
# apt-get update
# apt install -y lxd
# lxd init
# lxc launch images:ubuntu/trusty/amd64 router-xorp
```

Definice rozhraní je důležitým rozhodnutím pro budoucí funkčnost směrovačů, které jsou součástí síťové topologie. Pro směrovače Xorp jsou jednotlivá rozhraní přidávány v režimu physical, který z důvodu bezpečné manipulace vnoří zařízení do vnitřní struktury kontejneru.

```
lxc config device add router-xorp eth1 nic name=eth1
nictype=physical parent=nazevRozhrani
lxc config device add router-xorp eth2 nic name=eth2
nictype=physical parent=nazevRozhrani
lxc config device add router-xorp eth3 nic name=eth3
nictype=physical parent=nazevRozhrani
```

Po vnoření všech rozhraní je potřeba kontejner z důvodu aktivace rozhraní restartovat a opětovně spustit. Pokud by nedošlo k restartu kontejneru, jednotlivá zařízeni by nemohla být aktivována a použita pro přenos.

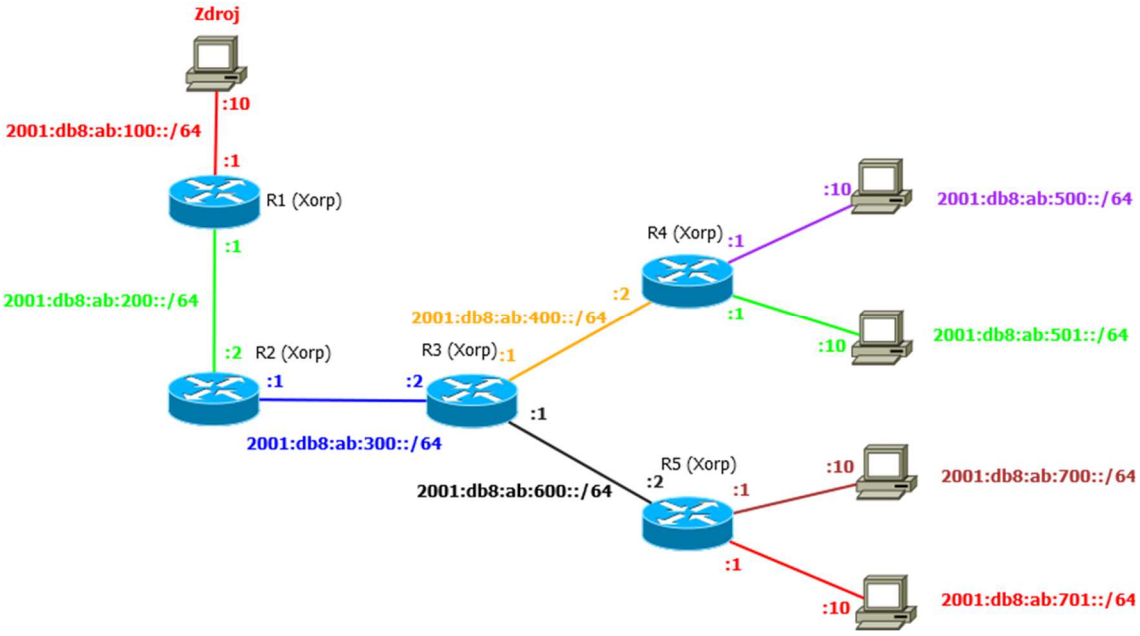
```
# lxc restart router-xorp
# lxc start router-xorp
```

V této fázi je virtuální kontejner připraven k použití a je možné vytvořit konfigurační soubor (soubor.boot) a soubor skript (skript.sh), díky kterým je dosaženo urychlení procesu vytvoření a spuštění směrovače. Soubory jsou následně vloženy do kontejneru.

```
# nano soubor.boot
# nano skript.sh
# lxc file push /home/student/xorp.ct-master.zip router-
xorp/home/ubuntu/
# lxc file push soubor.boot router-xorp/home/ubuntu/
# lxc file push skript.sh router-xorp/home/ubuntu/
# lxc shell router-xorp                    … připojení ke kontejneru
root@router-xorp:~# cd /home/ubuntu/
root@router-xorp:~# . skript.sh           … spuštění skriptu
```

5.2 Testování programu Xorp v režimu PIM-SM pro IPv6

Pro testování programu jsou vzhledem ke kompatibilitě systému Ubuntu a programu Xorp využity virtuální kontejnery LXD s Ubuntu 14.04 na směrovačích níže uvedené topologie. Topologie sítě se skládá z celkem pěti směrovačů, jednoho stream serveru a čtyř příjemců multicastu. K propojení veškerých zařízení v síti jsou využity Fast Ethernetové linky společně s Gigabit Ethernetovými rozhraními na směrovačích, které disponují protokoly OSPFv3, MLDv2 a PIM-SM.



Obrázek 5.1: Síťová topologie IPv6

Multicast v režimu Sparse Mode je specifický tím, že jeden ze směrovačů topologie je vybrán jako Rendezvous Point neboli shromaždiště. Tento směrovač by měl být vhodně zvolen, jelikož se jedná o místo setkání zdroje s příjemci a v topologii se jedná o rozhraní 2001:db8:ab:300::2 směrovače R3 (Xorp). Zvolením rozhraní lze přistoupit k určení multicastové adresy, která bude sloužit jednak pro zdroj multicastu jako adresa na kterou má data vysílat, ale i

pro příjemce jako adresa na které mohou data přijímat. Pro režim Sparse Mode jsou definovaná jasná pravidla, dle kterých je nutné adresu určit a po zvolení rozhraní 2001:db8:ab:300::2 jako RP je výsledná adresa ff7e:240:2001:db8:ab:300:0:20. K zajištění směrování je zvolen protokol OSPFv3 a pro registraci příjemců do skupiny protokol MLD ve verzi 2.

5.2.1 Výpisy směrovače R3 (Xorp) pro IPv6

Konfigurace a výpisy všech směrovačů, zdroje a příjemců síťové topologie jsou uvedeny v přílohách. Pokud je směrovač R3 (Xorp) zvolen jako RP je vhodné ověřit jeho výpisy a ověřit funkčnost celého přenosu. Pomocí příkazu show lze ze shellu samotného programu Xorp získat potřebné informace, které směrovač nabízí. Jedním z důležitých výpisů je ověření nastavení rozhraní pomocí příkazu show interfaces.

```
root@router-xorp> show interfaces eth1
eth1/eth1: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet6 2001:db8:ab:300::2 prefixlen 64
        inet6 fe80::5555 prefixlen 64
```

Z výstupu lze poznat korektní nastavení rozhraní eth1 na směrovači a podobným způsobem je možné ověřit i zbylé rozhraní. Dále je vhodné ověřit povolení přesměrování multicastu.

```
root@router-xorp> show mfea6 interface
Interface    State    Vif/PifIndex Addr            Flags
eth1         UP               0/18 2001:db8:ab:300::2 MULTICAST
BROADCAST KERN_UP
eth2         UP               1/19 2001:db8:ab:400::1 MULTICAST
BROADCAST KERN_UP
eth3         UP               2/20 2001:db8:ab:600::1 MULTICAST
BROADCAST KERN_UP
```

Přesměrování multicastu musí být povoleno nejen na samotných rozhraních, ale také v kernelu operačního systému. Pomocí výpisu show mfea6 interface je ověřen status těchto rozhraní a status povolení v kernelu operačního systému. Pokud by v kernelu operačního systému nedošlo nebo nebylo povoleno přesměrování multicastu, nebylo by možné doručovat data k příjemcům skrze síť.

Součástí multicastového přenosu je i využití protokolů, které slouží pro jeho realizaci. V testování jsou využity celkem tři protokoly OSPFv3, MLDv2 a PIM v režimu Sparse Mode. Konfigurace protokolů je součástí konfiguračního souboru a není nutné tyto informace nastavovat ručně. Dle níže uvedeného výpisu je ověřena funkčnost protokolu MLDv2 na všech připojených rozhraních a použitá verze.

```
root@router-xorp> show mld interface

Interface    State    Querier         Timeout Version Groups
eth1         UP       fe80::4444          164       2      13
eth2         UP       fe80::6666          175       2      13
eth3         UP       fe80::1235          196       2      13
```

Asi nejdůležitějším protokolem je protokol PIM. Protokol PIM zaručuje správné řízení dat, umístění příjemců a tvorbu distribučních stromů. V níže uvedeném výpisu je možné vyčíst status jednotlivých rozhraní, mód protokolu, ve kterém je nastaven nebo aktuálně použitou verzi protokolu.

```
root@router-xorp> show pim6 interface

Interface    State    Mode   V PIMstate Priority DRaddr
Neighbors
eth1         UP       Sparse 2 DR              1 fe80::5555
1
eth2         UP       Sparse 2 DR              1 fe80::8888
1
eth3         UP       Sparse 2 DR              1 fe80::7777
1
```

Jelikož je součástí třetího směrovače Rendezvous Point, nabízí shell tohoto směrovače možnost výpisu všech RP, které obsahuje. V případě směrovače R3 (Xorp) je obsažen pouze jediný bod na zvoleném rozhraní 2001:db8:ab:300::2, jehož volba byla provedena staticky.

```
root@router-xorp> show pim6 rps

RP           Type       Pri Holdtime Timeout ActiveGroups
GroupPrefix
2001:db8:ab:300::2 static    192       -1      -1            2
ff00::/8
```

Jako posledním protokolem je OSPFv3. OSPFv3 je použit na všech směrovačích, ale shell programu Xorp poskytuje omezenější možnosti výpisu protokolu. Pomocí show ospf6 neighbor lze zobrazit všechny sousední směrovače s hodnotou jejich ID či Link Local adresy.

```
root@router-xorp> show ospf6 neighbor

  Address          Interface              State      ID
fe80::4444          eth1/eth1               Full      20.20.20.20
fe80::8888          eth2/eth2               Full      40.40.40.40
fe80::1235          eth3/eth3               Full      50.50.50.50
```

5.2.2 Vyhodnocení funkčnosti multicastu v programu Xorp pro IPv6

I když je z pohledu výpisů programu Xorp veškeré nastavení v pořádku, nedospěla tato část testování k úspěšnému přenosu. Po podrobné kontrole všech konfiguračních souborů a výpisů, které nevykazovaly žádné chybné hlášení či chybnou syntaxi, bylo potřeba stanovit příčinu neúspěchu.

Po dosažení kompatibility distribuce Ubuntu 14.04 s programem Xorp 1.8.6 pomocí virtuálních kontejnerů LXD a správného nastavení konfiguračních souborů, bylo nutné ověřit, zda příslušný kernel operačního systému IPv6 multicast podporuje. Toto podezření bylo potvrzeno. Vzhledem k tomu, že konfigurační soubor jádra operačního systému obsahuje veškeré parametry spojené s funkčností multicastu v IPv6.

```
CONFIG_IPV6_MROUTE=y
CONFIG_IPV6_PIMSM_V2=y
```

Další možností nepříznivé funkcionality programu je jeho neschopnost nastavit přesměrování multicastu. Z výpisu shellu jednotlivých směrovačů Xorp je patrné, že nastavení proběhlo bez jakýchkoliv závad, avšak pro definitivní potvrzení této schopnosti je doložen výpis ze směrovače R3 (Xorp) příkazem sysctl.

```
root@router-xorp:~# sysctl -a | grep ipv6 | grep forward
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.all.mc_forwarding = 1
net.ipv6.conf.eth1.forwarding = 1
net.ipv6.conf.eth1.mc_forwarding = 1
net.ipv6.conf.eth2.forwarding = 1
net.ipv6.conf.eth2.mc_forwarding = 1
net.ipv6.conf.eth3.forwarding = 1
net.ipv6.conf.eth3.mc_forwarding = 1
```

Po delším zkoumání a čerpání informací z různých zdrojů, ať už knižních či internetových, se vyskytla možnost souvislosti nekorektní funkce programu s velikostí MTU (Maximum Transmission unit). Obvyklá hodnota MTU je stanovena na hodnotu 1500 bajtů a právě s touto hodnotou mohou mít vybrané verze kernelu potíže. Ovšem i po omezení velikosti hodnoty MTU, ať už v konfiguračním souboru či stream serveru nevedlo opatření k úspěchu.

Z pohledu systému jsou veškeré prvky pro funkčnost multicastu splněny a správně nastaveny. Zbývá tedy jediná možnost a také příčina problému, a tím je nekorektní chování samotného programu Xorp. Program Xorp je použit v nejnovější verzi 1.8.6 a jedná se o jedinou funkční verzi, kterou společnost vydala. Vzhledem k tomu, že všechny konfigurační soubory i výpisy jsou v pořádku je potřeba stanovit, v které části programu chyba nastává. Prvním krokem

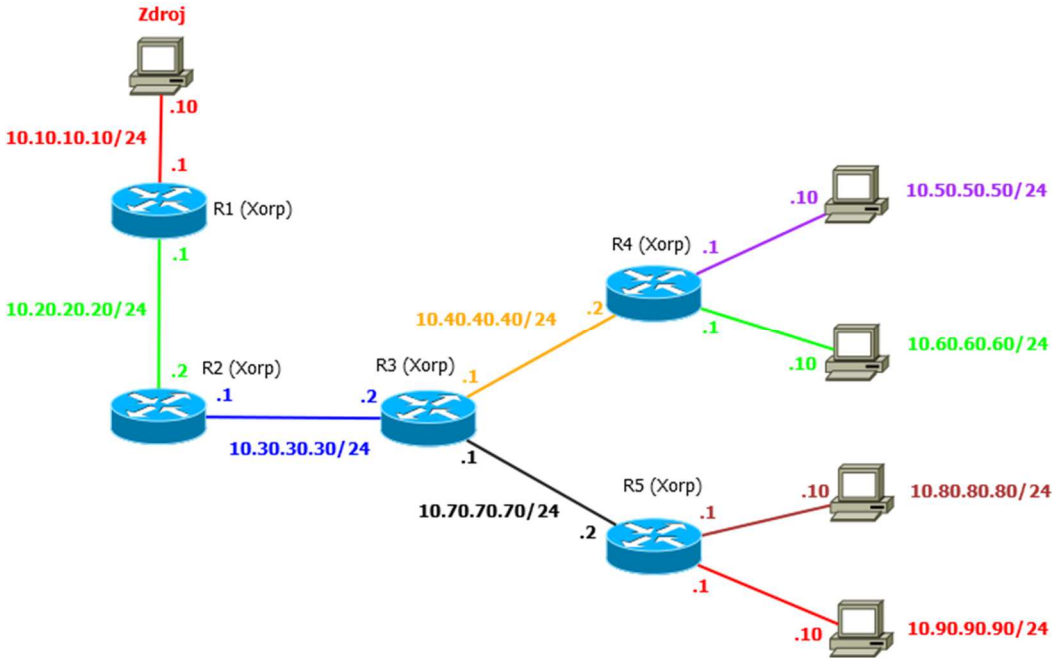
pro stanovení příčiny je zjistit, kam až ve skutečnosti data ze serveru dorazí. K ověření poslouží nástroj tcpdump.

```
root@router-xorp:/usr/local/xorp/sbin# tcpdump -i eth1
13:44:47.370112 IP6 2001:db8:ab:100::10.54195 >
ff7e:240:2001:db8:ab:300:0:20.1234: UDP, length 1316
13:44:47.371004 IP6 2001:db8:ab:100::10.54195 >
ff7e:240:2001:db8:ab:300:0:20.1234: UDP, length 1316
13:44:47.371005 IP6 2001:db8:ab:100::10.54195 >
ff7e:240:2001:db8:ab:300:0:20.1234: UDP, length 1316
root@router-xorp:/usr/local/xorp/sbin# tcpdump -i eth2
13:46:14.891664 IP6 fe80::3333 > ff02::5: OSPFv3, Hello, length
40
13:46:22.591819 IP6 fe80::2222 > ff02::d: PIMv2, Hello, length
56
13:46:23.014573 IP6 fe80::2222 > ff02::5: OSPFv3, Hello, length
40
```

Výše uvedené výpisy programu tcpdump jsou získané z prvního směrovače R1 (Xorp). Zde je patrné, že multicastová data sice přicházejí na rozhraní eth1, ale z rozhraní eth2 jsou vysílány pouze zprávy typu Hello. Multicastová data se tak nedostanou přes první směrovač a tím je odhaleno jádro problému. Problém může souviset se špatnou funkčností protokolu PIM pro IPv6 nebo protokolu MLDv2. Úkolem protokolu MLDv2 je registrace příjemců do multicastových skupin. Vzhledem k výpisům směrovačů, ke kterým jsou příjemci připojení a které vykazují jejich přítomnost a snahu se registrovat, nelze protokol MLDv2 považovat za chybnou součást. Tou je protokol PIM, který slouží pro IPv6. Protokol PIM pro IPv6 není schopen generovat tzv. Register zprávu, kterou by měl první směrovač v síti R1 (Xorp) poslat směrem k RP a informovat tak o přítomnosti zdroje vysílajícího multicastová data. Směrovač R3 (Xorp) na kterém je RP přítomen o zdroji multicastu neví a nedochází k distribuci dat skrze počítačovou síť. Tomuto faktu napomáhá také skutečnost, že poslední úpravy a modifikace programu Xorp jsou datovány k roku 2012.

5.3 Testování programu Xorp v režimu PIM-SM pro IPv4

Vzhledem k nefunkčnosti multicastu v programu Xorp pro IPv6 je v rámci testování vyzkoušen multicast v režimu PIM-SM pro IPv4. Topologie sítě je totožná s topologií, která je využita pro IPv6 s novým adresováním a novou multicastovou adresou. Jelikož adresování multicastu v sítích IPv4 má předem daný rozsah, není potřeba jako u IPv6 multicastovou adresu počítat, ale pouze vybrat z přiděleného rozsahu 224.0.0.0 až 239.255.255.255. K testování je zvolena adresa 239.1.1.1 za podpory protokolů IGMPv3, OSPFv2 a PIM v režimu Sparse Mode.



Obrázek 5.2: Síťová topologie IPv4

5.3.1 Výpisy směrovače R3 (Xorp) pro IPv4

V síťové topologii pro IPv4 je směrovač R3 (Xorp) opětovně zvolen jako RP a proto je vhodné analyzovat jeho výpisy. Prvním je výpis show interfaces pro kontrolu správného nastavení jednotlivých rozhraní. V níže uvedeném výpisu jsou uvedeny informace týkající se rozhraní eth1 a podobným způsobem lze kontrolovat i zbylé připojené rozhraní.

```
root@router-xorp> show interfaces eth1

eth1/eth1: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps

inet 10.30.30.2 subnet 10.30.30.0/24 broadcast 10.30.30.255
```

Po ověření korektnosti nastavení všech rozhraní je nutné ověřit povolení přesměrování multicastu. Zde je velmi důležité, aby jeho povolení proběhlo správně nejen na jednotlivých rozhraních, ale také v kernelu operačního systému.

```
root@router-xorp> show mfea interface

Interface     State     Vif/PifIndex Addr            Flags
eth1          UP                0/6 10.30.30.2     MULTICAST
BROADCAST KERN_UP

eth2          UP                1/7 10.40.40.1     MULTICAST
BROADCAST KERN_UP

eth3          UP                2/8 10.70.70.1     MULTICAST
BROADCAST KERN_UP
```

Z výpisu show mfea interfaces je zřejmé, že veškeré povolení přesměrování proběhlo v pořádku, včetně povolení v kernelu a lze tak přikročit ke kontrole protokolů. Prvním z nich je protokol IGMPv3. Jedná se prakticky o totožný protokol v porovnání s MLDv2, avšak jeho funkčnost je omezena výhradně v rámci IPv4 sítí.

```
root@router-xorp> show igmp interface

Interface    State    Querier         Timeout Version Groups
eth1         UP       10.30.30.1          163       3      5
eth2         UP       10.40.40.1         None       3      5
eth3         UP       10.70.70.1         None       3      5
```

Aktivace protokolu na jednotlivých rozhraních dle výše uvedeného výpisu show igmp interface byla vykonána správně a je tak možné přikročit k protokolu PIM. Protokol je opět definován v režimu Sparse Mode a nejnovější verzi 2. Pomocí příkazu show pim interface lze nastavení ověřit.

```
root@router-xorp> show pim interface

Interface    State    Mode   V PIMstate Priority DRaddr
Neighbors
eth1         UP       Sparse 2 DR              1 10.30.30.2
1
eth2         UP       Sparse 2 NotDR           1 10.40.40.2
1
eth3         UP       Sparse 2 NotDR           1 10.70.70.2
1
```

Díky tomu, že je součástí směrovače R3 (Xorp) shromaždiště neboli Rendezvous Point, nabízí shell programu Xorp možnost jeho kontroly pomocí příkazu show pim rps. RP je zvolen staticky na jednom rozhraní směrovače a tím je rozhraní 10.30.30.2. Z výpisu je možné vyčíst adresu zvoleného rozhraní včetně povolených multicastových skupin.

```
root@router-xorp> show pim rps

RP              Type      Pri Holdtime Timeout ActiveGroups
GroupPrefix
10.30.30.2      static    192       -1      -1            1
224.0.0.0/4
```

Posledním protokolem, který je součástí testování je protokol OSPFv2 pro zajištění směrování v síťové topologii. OSPF je využit ve verzi 2, protože tato verze je určená výhradně ke směrování v IPv4 sítích a nikoli pro IPv6. S ohledem na omezené možnosti výpisu protokolu ze shellu programu Xorp je možné kontrolovat pouze sousední směrovače nebo OSPF databázi. K výpisu sousedních směrovačů slouží příkaz show ospf4 neighbor.

```
root@router-xorp> show ospf4 neighbor
  Address         Interface              State     ID
10.30.30.1        eth1/eth1              Full      20.20.20.20
10.40.40.2        eth2/eth2              Full      40.40.40.40
10.70.70.2        eth3/eth3              Full      50.50.50.50
```

5.3.2 Vyhodnocení funkčnosti multicastu v programu Xorp pro IPv4

Chybnou funkčnosti multicastu v IPv6 je do testování zahrnut příklad pro multicastový přenos, který je realizován v IPv4 síti. Výsledkem přenosu je stabilní stream, jehož funkčnost je potvrzena nejen úspěšným přenosem, ale také korektností výpisů ze shellu programu Xorp a kernelu systému.

```
CONFIG_IP_MULTICAST=y
CONFIG_IP_MROUTE=y
CONFIG_IP_PIMSM_V2=y
```

Povolení přesměrování multicastu je z výpis směrovače R3 (Xorp) v pořádku a podobně jako u IPv6 i zde je kompatibilita programu Xorp 1.8.6 a operačního systému dosažena díky virtuálním kontejnerům LXD.

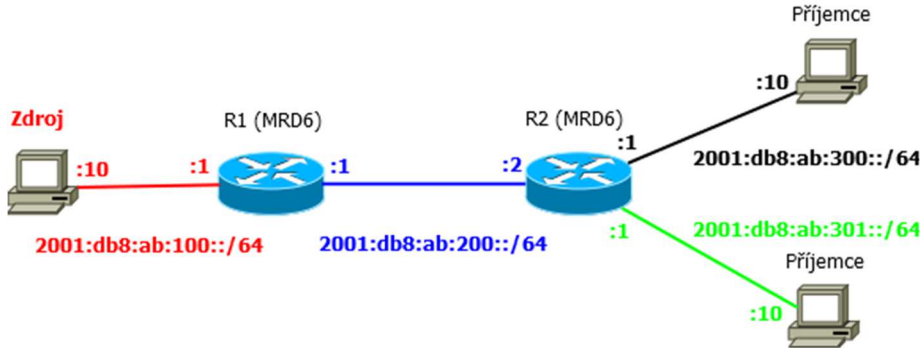
```
root@router-xorp:~# sysctl -a | grep ipv4 | grep forward
net.ipv4.conf.all.forwarding = 1
net.ipv4.conf.all.mc_forwarding = 1
net.ipv4.conf.eth1.forwarding = 1
net.ipv4.conf.eth1.mc_forwarding = 1
net.ipv4.conf.eth2.forwarding = 1
net.ipv4.conf.eth2.mc_forwarding = 1
net.ipv4.conf.eth3.forwarding = 1
net.ipv4.conf.eth3.mc_forwarding = 1
```

Závěrem testování je výpis programu tcpdump jednoho z příjemců, který o multicastový přenos požádal prostřednictvím programu VLC Media Player, čímž je příjem dat potvrzen.

```
root@pc14:~# tcpdump -i enx00ed4d680638
14:10:17.950749 IP 10.10.10.10.39904 > 239.1.1.1.1234: UDP,
length 1316
14:10:17.950777 IP 10.10.10.10.39904 > 239.1.1.1.1234: UDP,
length 1316
```

5.4 Testování programu MRD6

Druhým programem k testování multicastového přenosu je zvolen program MRD6. Pro tuto část testu je vytvořena nová síťová topologie s využitím dvou směrovačů, jednoho zdroje a dvou příjemců s odpovídajícím adresováním. V porovnání s programem Xorp, který je realizován v Ubuntu 14.04 je zde využit původní systém Ubuntu 18.04 bez virtuálních kontejnerů pro zajištění kompatibility.



Obrázek 5.3: Síťová topologie IPv6 pro MRD6

Na jednotlivých směrovačích je naistalován program MRD6 a společně s konfiguračním souborem v adresáři /etc/mrd6.conf tvoří základ pro realizaci multicastového přenosu. Konfigurační soubor je jediný, který je potřeba upravit a nastavit s parametry pro protokoly PIM a MLDv2. Co se týče směrování v síti, pak celá topologie je nakonfigurována staticky a není zde využitý žádný směrovací protokol. Pokud dochází ke změnám v konfiguračním souboru je vhodné službu před testováním restartovat a zjistit její status nebo chybové hlášení pomocí příkazu service mrd6 status.

```
root@pc13: ~
File  Edit  View  Search  Terminal  Help
root@pc13:~# service mrd6 status
● mrd6.service - LSB: IPv6 Multicast Routing Daemon
   Loaded: loaded (/etc/init.d/mrd6; generated)
   Active: active (running) since Tue 2019-03-05 12:05:12 CET; 46min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 4478 ExecStop=/etc/init.d/mrd6 stop (code=exited, status=0/SUCCESS)
  Process: 4484 ExecStart=/etc/init.d/mrd6 start (code=exited, status=0/SUCCESS)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/mrd6.service
           └─4489 /usr/sbin/mrd6 -D

Mar 05 12:05:12 pc13 systemd[1]: Stopped LSB: IPv6 Multicast Routing Daemon.
Mar 05 12:05:12 pc13 systemd[1]: Starting LSB: IPv6 Multicast Routing Daemon...
Mar 05 12:05:12 pc13 mrd6[4478]:  * Stopping Multicast routing daemon: mrd6
Mar 05 12:05:12 pc13 mrd6[4478]:    ...done.
Mar 05 12:05:12 pc13 systemd[1]: Started LSB: IPv6 Multicast Routing Daemon.
Mar 05 12:05:12 pc13 mrd6[4484]:  * Starting Multicast routing daemon: mrd6
Mar 05 12:05:12 pc13 mrd6[4484]:    ...done.
root@pc13:~#
```

Obrázek 5.4: Status služby MRD6

5.4.1 Konfigurace směrovače R1 (MRD6)

Konfigurační soubor obou směrovačů, který je umístěn v adresáři /etc/mrd6.conf má velmi podobnou strukturu a nevyžaduje rozsáhlé úpravy pro konečnou fázi testování. Soubor obsahuje tzv. moduly. Moduly odpovídají jednotlivým funkcím a protokolům, které slouží pro realizaci multicastového přenosu a musí být součástí konfiguračního souboru.

```
load-module mld;
load-module pim;
```

Po načtení modulů je možné přikročit k definici nastavení protokolu PIM. U protokolu MLDv2 není nutné definovat žádné rozhraní či nastavení, jelikož program dokáže funkci protokolu aplikovat automaticky na všechny rozhraní.

```
/* Global pim variable configuration */
pim {
        /* we want to be a BSR candidate */
        enable bsr-candidate;
}
/* Groups configuration */
groups {
        /* group mask */
        ff7e:240:2001:db8:ab:200::/96 {
                pim rp 2001:db8:ab:200::2;
        }
}
```

Součástí konfigurace PIM je definice skupiny pro příjem a také adresa RP. Oproti jiným konfiguracím je přítomna funkce bsr-candidate. Funkce bsr-candidate slouží k automatickému vyhledání dostupných RP v síti a veškeré informace, které zaznamená shromažďuje a aplikuje v síti dle potřeby.

5.4.2 Výpisy směrovače R1 (MRD6)

Pro výpis z jednotlivých směrovačů nabízí program MRD6 možnost získat informace ze shellu samotného programu a zjistit tak nastavení skupin a rozhraní. Informace o nastavení rozhraní lze získat pomocí příkazu mrd6sh show interface. Výpis obsahuje veškeré informace týkající se nastavení všech rozhraní, které jsou připojeny k zařízení. Je možné vyčíst např. název, status a IP adresu. Podstatnější informace jsou o protokolech MLDv2 a PIM. Podle těchto informací, lze určit, zda jsou na jednotlivých rozhraních aktivní a zda mají sousední směrovače s podporou stejných protokolů.

```
root@pc14:~# mrd6sh show interface
Interface enx00ec4d68068a (3) is Up (Uptime: 17m 53s)
  Global: 2001:db8:ab:100::1/64 <PRIMARY>
  MLD, version 2
    Querier: self
  PIM
    Neighbours:
      (None)
Interface enx00ec4c680637 (4) is Up (Uptime: 17m 53s)
  Global: 2001:db8:ab:200::1/64 <PRIMARY>
  MLD, version 2
    Querier: fe80::1111 for 2m 54s
  PIM
    Neighbours:
      fe80::1111, 1m 42s
        Secundary-Addresses:
          2001:db8:ab:200::2
```

Informace o multicastových skupinách, které jsou součástí konfiguračního souboru, lze získat prostřednictvím příkazu mrd6sh show group. Tyto informace jednoznačně definují povolené multicastové adresy, včetně adresy RP a také zdroje.

```
root@pc14:~# mrd6sh show group
Group ff7e:240:2001:db8:ab:200:0:20
  PIM
    Rendezvous-Point: 2001:db8:ab:200::2 [embedded]
    Sources:
      (2001:db8:ab:100::10), SPT, Active, Uptime: 18m 20s
        Register-Stop: 20s
        Input Interface: enx00ec4d68068a, Upstream: (Local)
```

5.4.3 Konfigurace směrovače R2 (MRD6)

Konfigurace druhého směrovače je prakticky totožná s konfigurací směrovače R1 (MRD6). Výjimkou je nastavení protokolu PIM, který musí obsahovat funkci rp-candidate. Jedná se o funkci, jejímž úkolem je propagace zařízení jako kandidáta na RP.

```
/* Global pim variable configuration */
pim {
        /* we want to be a BSR candidate */
        enable bsr-candidate;
        /* we want to be a RP candidate */
        enable rp-candidate;
}
```

5.4.4 Výpisy směrovače R2 (MRD6)

Výpisy související s rozhraními směrovače R2 (MRD6) jsou stejného charakteru jako směrovač R1 (MRD6). Co je podstatné a v čem se zařízení liší je výpis skupin. Vzhledem k tomu, že součástí směrovače je rozhraní s totožnou adresou jako RP, přibyl v protokolu PIM příznak self. Příznak self oznamuje, že aktuální směrovač je sám vyhrazen jako RP s čímž musí počítat funkce bsr-candidate.

```
root@pc13:~# mrd6sh show group
Group ff7e:240:2001:db8:ab:200:0:20
  PIM
    Rendezvous-Point: 2001:db8:ab:200::2 [embedded, self]
    Sources:
      (2001:db8:ab:100::10), Active, Uptime: 38m 54s
        Input Interface: enx00133ba00429
```

5.4.5 Vyhodnocení funkčnosti multicastu v programu MRD6

U programu MRD6 jsou otázky instalace a nastavení konfiguračních souborů hodnoceny velmi kladně a nejedná se o složitý postup. Avšak funkcionalita programu je pravým opakem a přenos multicastových dat pomocí MRD6 nedosáhl úspěchu. Vzhledem k této skutečnosti je nutné podobně jako u programu Xorp pro IPv6 stanovit příčinu špatné funkce.

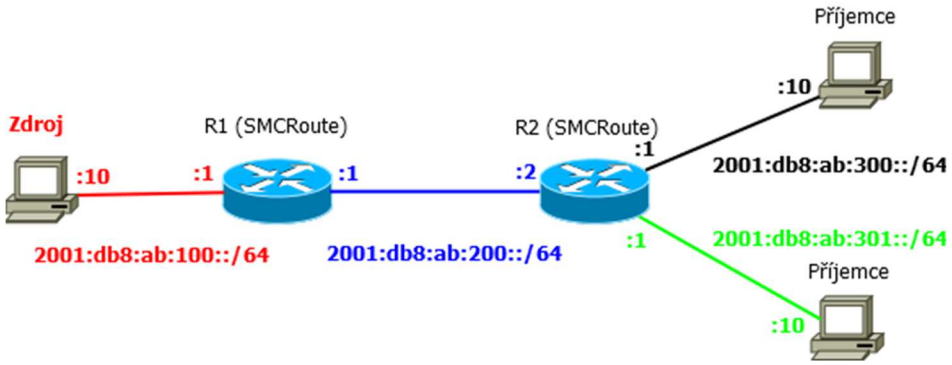
Z pohledu nastavení kernelu systému a konfiguračních souborů jsou veškeré informace a výpisy v pořádku. Co program ovšem nedokáže změnit je přesměrování multicastu. Za pomocí příkazu sysctl je zjištěno, že program MRD6 není schopen nastavit přesměrování multicastu v systému a díky tomu směrovače nedokáží data přeposílat.

```
root@pc13:~# sysctl -a | grep ipv6 | grep forward
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.all.mc_forwarding = 0
```

Zjištěním problému s přesměrováním byly vyzkoušeny různé způsoby manuálního povolení. Nicméně i pokusy o povolení nepomohly k povolení parametru mc_forwarding a tato část testování nedosáhla úspěšného přenosu.

5.5 Testování programu SMCRoute

Posledním testovaným programem v práci je program SMCRoute. Program SMCRoute je podobně jako MRD6 testován ve stejném systému Ubuntu 18.04 a stejné topologii, která se skládá z jednoho zdroje multicastu, dvou směrovačů a příjemců.



Obrázek 5.5: Síťová topologie IPv6 pro SMCRoute

Směrovače obsahují program SMCRoute a konfigurační soubor v adresáři /etc/smcroute.conf. Jedná se opět o jediný soubor, který je zodpovědný za přenos multicastových dat skrze síť a který je nutné upravit. Směrování je realizováno nikoli pomocí směrovacího protokolu, ale staticky na jednotlivých strojích. Po zajištění směrování, instalaci programu a úpravě konfiguračního souboru je potřeba službu SMCRoute restartovat a ověřit její status nebo chybová hlášení.

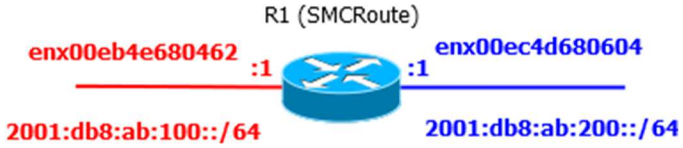
```
root@pc2: ~
File  Edit  View  Search  Terminal  Help
root@pc2:~# service smcroute status
● smcroute.service - LSB: Static multicast router daemon
   Loaded: loaded (/etc/init.d/smcroute; generated)
   Active: active (running) since Mon 2019-03-04 14:13:25 CET; 5s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 3910 ExecStop=/etc/init.d/smcroute stop (code=exited, status=0/SUCCESS)
  Process: 3916 ExecStart=/etc/init.d/smcroute start (code=exited, status=0/SUCCESS)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/smcroute.service
           └─3921 /usr/sbin/smcroute -d

Mar 04 14:13:25 pc2 systemd[1]: Stopped LSB: Static multicast router daemon.
Mar 04 14:13:25 pc2 smcroute[3910]:  * Stopping static multicast router daemon: smcroute
Mar 04 14:13:25 pc2 smcroute[3910]:    ...done.
Mar 04 14:13:25 pc2 systemd[1]: Starting LSB: Static multicast router daemon...
Mar 04 14:13:25 pc2 smcroute[3921]: SMCRoute version 2.0.0
Mar 04 14:13:25 pc2 smcroute[3916]:  * Starting static multicast router daemon: smcroute
Mar 04 14:13:25 pc2 smcroute[3916]:    ...done.
Mar 04 14:13:25 pc2 systemd[1]: Started LSB: Static multicast router daemon.
root@pc2:~#
```

Obrázek 5.6: Status služby SMCRoute

5.5.1 Konfigurace směrovače R1 (SMCRoute)

Konfigurační soubor programu SMCRoute obsahuje základní ukázky konfigurací multicastového přenosu. Pro zprovoznění multicastu v IPv6 stačí upravit jednu z těchto konfigurací a nahradit názvy rozhraní včetně adres.

```
mgroup from nazevVstupnihoRozhrani(enx00eb4e680462) group
ff3e::20

mroute from nazevVstupnihoRozhrani(enx00eb4e680462) group
ff3e::20 source IPv6_adresa_zdroje(2001:db8:ab:100::10) to
nazevVystupnihoRozhrani(enx00ec4d680604)
```

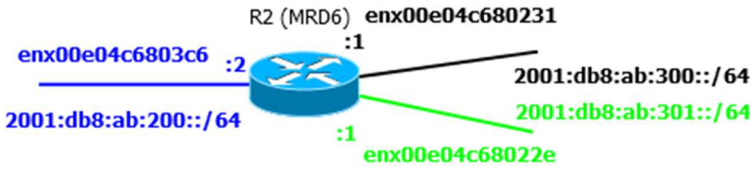


Obrázek 5.7: Rozhraní směrovače R1 (SMCRoute)

5.5.2 Konfigurace směrovače R2 (SMCRoute)

Nastavení druhého směrovače R2 (SMCRoute) je stejné charakteru s použitím odlišných názvů rozhraní.

```
mgroup from nazevVstupnihoRozhrani(enx00e04c6803c6) group
ff3e::20

mroute from nazevVstupnihoRozhrani(enx00e04c6803c6) group
ff3e::20 source IPv6_adresa_zdroje(2001:db8:ab:100::10) to
nazevVystupnichRozhrani(enx00e04c680231 enx00e04c68022e)
```



Obrázek 5.8: Rozhraní směrovače R2 (SMCRoute)

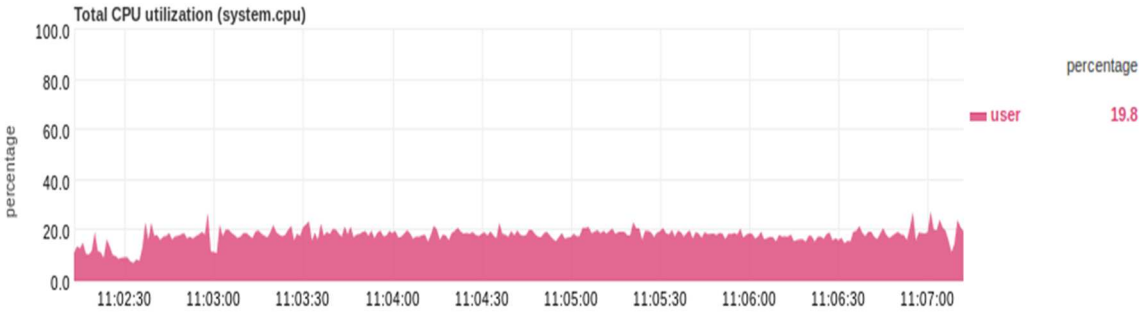
5.5.3 Vyhodnocení funkčnosti multicastu v programu SMCRoute

Program SMCRoute je velmi kompaktní a jednoduchý program určený pro distribuci multicastu. Po úpravě konfiguračních souborů a restartování programu na vlastní síťové topologii došlo k úspěšnému přenosu multicastových dat a nalezení tak velmi jednoduchého způsobu, jak distribuovat multicastová data skrze počítačovou síť. Pro obohacení testování je pomocí programu netstat sledováno využití procesoru, paměti RAM a rozhraní stream serveru při využití tří druhů kodeků. Mezi tyto kodeky jsou zařazeny H.265, H.264 a MPEG-4 Video.

5.5.4 Využití procesoru serveru pro kodeky H.265, H.264 a MPEG-4 Video

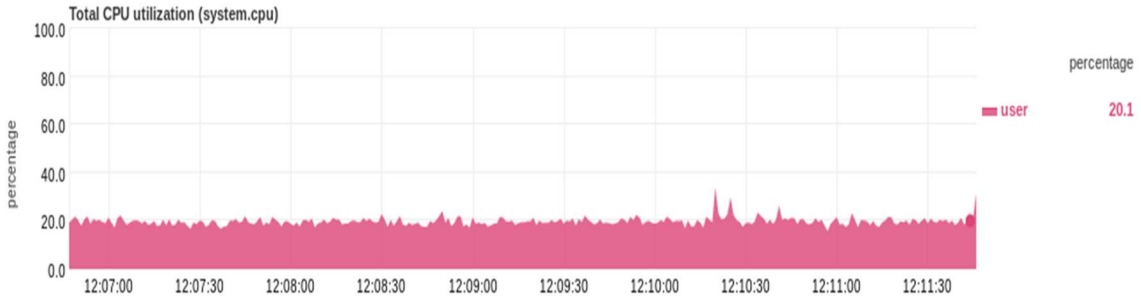
Kodeky slouží pro kompresi dat, které lze přenášet ze zdroje směrem k uživateli. K této kompresi je zapotřebí systémových prostředků jako např. procesoru nebo operační paměti. Jelikož se každý kodek odlišuje v tom, jak přenášená data zpracovává, odlišuje se také množstvím systémových prostředků, které využívá ke své činnosti. Mezi testované kodeky jsou vybrány kodeky H.265, H.264 a MPEG-4 Video. Jedná se o nejpopulárnější typy, které je možné využít v oblasti streamování, videokonferencí nebo televizního vysílání.

Prvním testovaným kodekem v příkladu SMCRoute je kodek H.264 s rychlostí 2000 kbit/s za podpory kodeku MP3 pro audio data s rychlostí 320 kbit/s. Z obrázku 5.9 lze vyčíst procentuální vytížení procesoru sloužící pro zpracování streamovaných dat. Samotný stream s kombinací těchto dvou kodeků dosahuje tedy přibližně 20 % využití celkové kapacity procesoru.



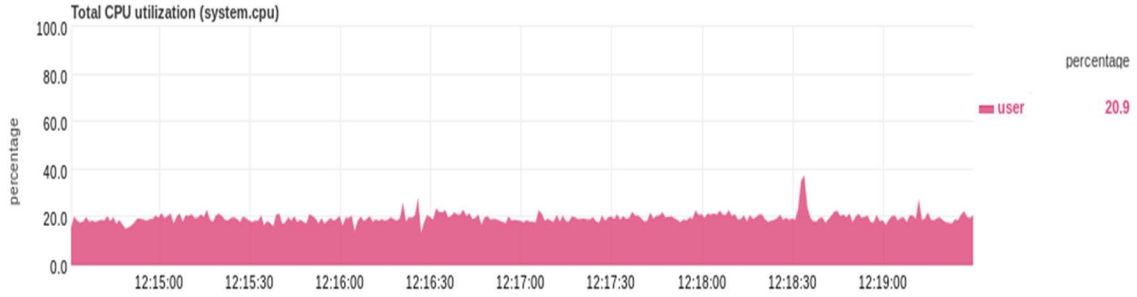
Obrázek 5.9: Využití procesoru s kodekem H.264 pro 2000 kbit/s

Pro důkladnější testování kodeku H.264 a jeho efektivity je podroben situaci, kdy dojde k navýšení přenosových rychlostí video kodeku. Hodnota 2000 kbit/s není příliš vysoká, proto jsou zvoleny rychlosti 10 000 a 20 000 kbit/s. Obrázek 5.10 vyjadřuje vytížení procesovu stream serveru s rychlostí 10 000 kbit/s.



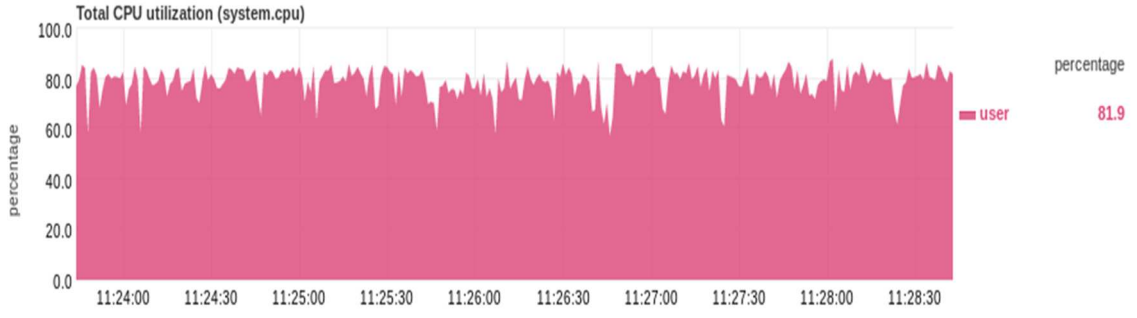
Obrázek 5.10: Využití procesoru s kodekem H.264 pro 10 000 kbit/s

I s pětinásobným nárustem rychlosti je tento kodek k systémovým prostředkům velice šetrný a dochází k minimálnímu nárustu využití procesoru. Vzhledem k dobrým výsledkům je kodek využit pro ještě vyšší rychlost 20 000 kbit/s. Z obrázku 5.11 je zřejmé, že i tato rychlost není pro H.264 komplikací a dosahuje stále velmi nízkých hodnot využití. V porovnání s rychlostí 2000 kbit/s je nárust výkonu v jednotkách procent.



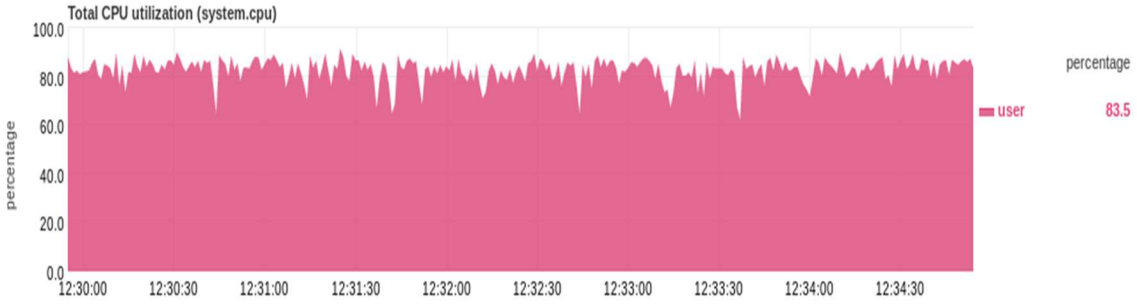
Obrázek 5.11: Využití procesoru s kodekem H.264 pro 20 000 kbit/s

Vytížení procesoru u kodeku H.264 není tak významné jako v případě jiných. Kodek je převážně využíván pro standard DVB-T a postupem času má být nahrazen novějším H.265 pro standard DVB-T2. Novější kodek H.265 je znám svou výpočetní náročností, což je potvrzeno i z výsledku přenosu multicastových dat s totožnými rychlostmi jako u předchozího testování.



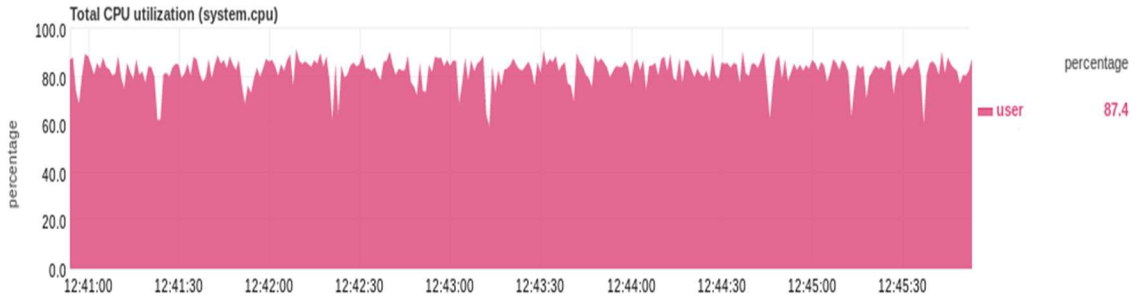
Obrázek 5.12: Využití procesoru s kodekem H.265 pro 2000 kbit/s

Využití procesoru pro samotný stream s kodekem H.265 (2000 kbit/s) a MP3 (320 kbit/s) bez výrazných potíží atakuje hranici 80 %. Dosahuje až čtyřikrát vyšších hodnot v porovnání s kodekem H.264, tím nepříznivě ovlivňuje i stabilitu streamu. Vyšší přenosové rychlosti u tohoto druhu není vhodné používat, protože lze předpokládat také nárust systémových prostředků.



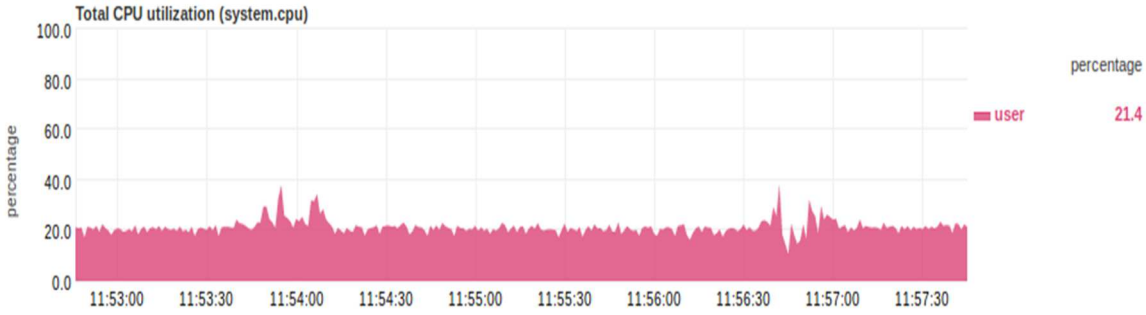
Obrázek 5.13: Využití procesoru s kodekem H.265 pro 10 000 kbit/s

Pro 10 000 kbit/s je vyžadován stále vyšší výkon procesoru. Procesor je nucen udržovat své vytížení na hodnotách přes 80 % a překonává výsledky z měření pro 2000 kbit/s. Situace pro nejvyšší rychlost není jiná a jsou překonány také hodnoty pro 10 000 kbit/s.



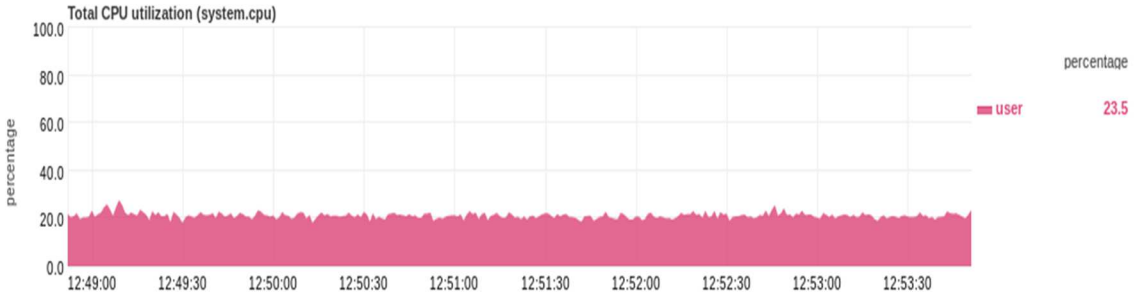
Obrázek 5.14: Využití procesoru s kodekem H.265 pro 20 000 kbit/s

Posledním z využitých kodeků pro testování je MPEG-4 Video s rychlostí 2000 kbit/s a MP3 s rychlostí 320 kbit/s. MPEG-4 Video je v oblasti streamu velmi známý a slouží např. pro videokonference nebo televizní přenos. Hodnotami využití procesoru s rychlostí 2000 kbit/s se velmi podobá kodeku H.264, ale vzhledem k výsledkům při vyšších přenosových rychlostech není MPEG-4 Video k systémovým prostředkům natolik šetrný.



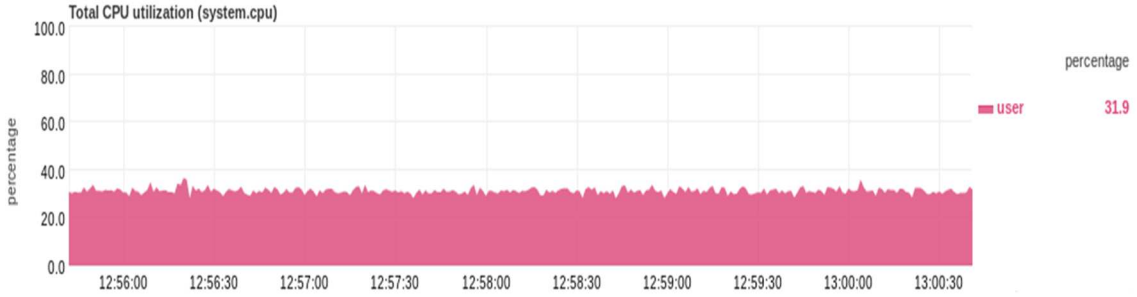
Obrázek 5.15: Využití procesoru s MPEG-4 Video pro 2000 kbit/s

Pro 10 000 kbit/s se rozdíl vytížení procesoru stále pohybuje v jednotkách procent a hodnoty nejsou od kodeku H.264 příliš vzdálené.



Obrázek 5.16: Využití procesoru s MPEG-4 Video pro 10 000 kbit/s

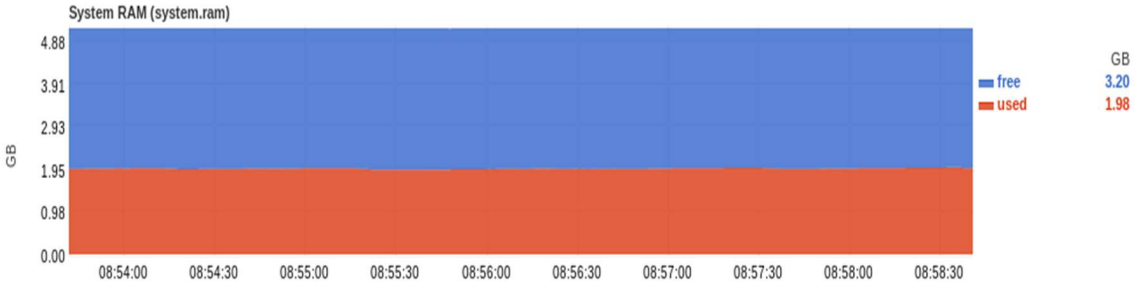
Tato skutečnost neplatí pro nejvyšší rychlost 20 000 kbit/s. Zde je vytížení procesoru navýšeno a přesahuje hodnoty 30 % celkového výkonu pro stream. Díky výsledným hodnotám všech kodeků lze říci, že z pohledku efektivního využití procesoru je nejvhodnější volbou kodek H.264. Dokáže pracovat i s vysokými přenosovými rychlosti s minimálním dopadem na vytížení procesoru.



Obrázek 5.17: Využití procesoru s MPEG-4 Video pro 20 000 kbit/s

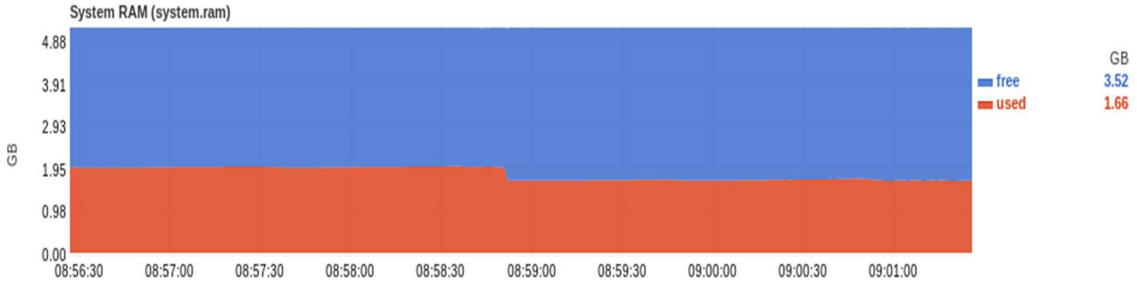
5.5.5 Využití RAM serveru pro kodeky H.265, H.264 a MPEG-4 Video

Operační paměť RAM je jeden z dalších systémových prostředků, který slouží pro zpracování streamovaného audiovizuálního obsahu. Pro každý video kodek v testování je zvolena pevná rychlost 2000 kbit/s za podpory audio kodeku MP3 s rychlostí 320 kbit/s. Rozdíl ve využití paměti pro vyšší rychlosti, není tak markantní, a tak nejsou tyto data sledována.



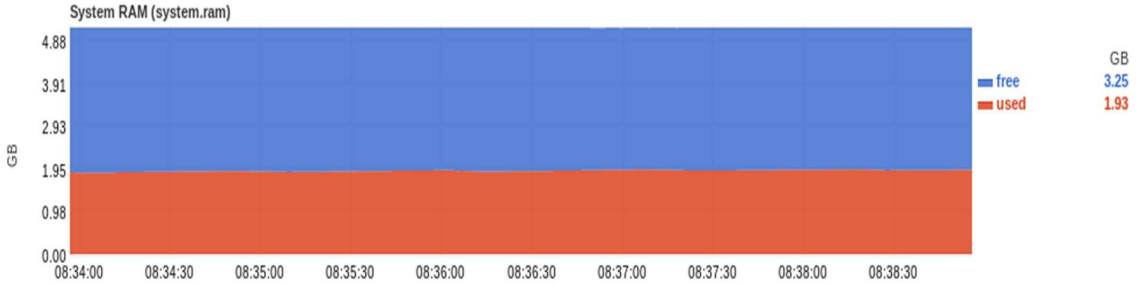
Obrázek 5.18: Využití RAM s aktivním streamem (H.264) 2000 kbit/s

Pro kodek H.264 (2000 kbit/s) a MP3 (320 kbit/s) je celkové využití operační paměti serveru s aktivním streamem přibližně 1,98 GB.



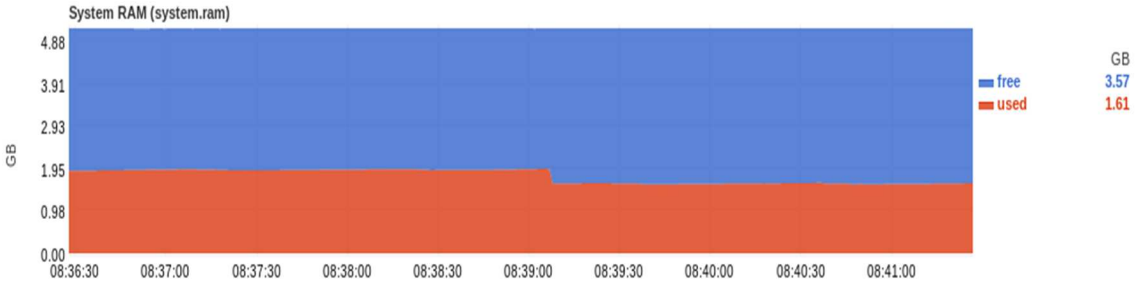
Obrázek 5.19: Využití RAM s neaktivním streamem (H.264) 2000 kbit/s

Po ukončení streamu dochází k postupnému uvolnění operační paměti. Tento proces je zřejmý z obrázku 5.19, kde nastává zlom a lze určit hodnotu paměti sloužící výhradně streamu. Celková hodnota využití paměti RAM pro stream s kodeky H.264 (2000 kbit/s) a MP3 (320 kbit/s) činí 0,32 GB.



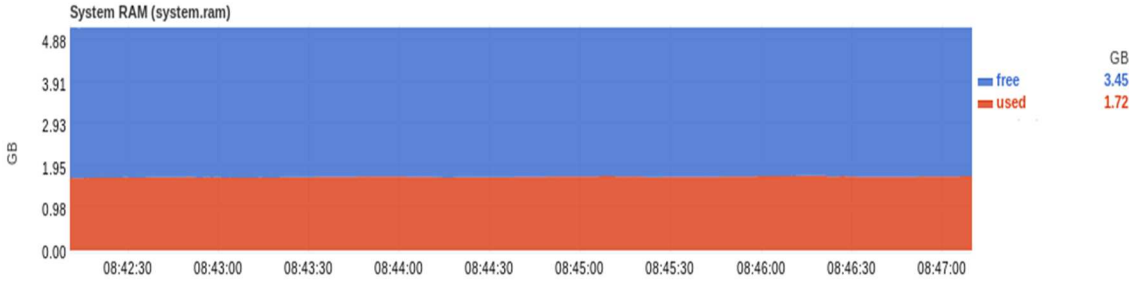
Obrázek 5.20: Využití RAM s aktivním streamem (H.265) 2000 kbit/s

Stream s kodekem H.265 vykazuje hodnotu celkové využité paměti serveru 1,93 GB.



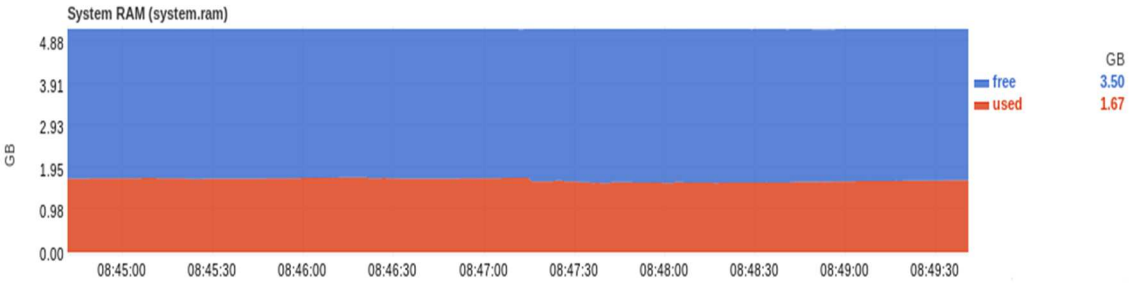
Obrázek 5.21: Využití RAM s neaktivním streamem (H.265) 2000 kbit/s

Ukončením streamu dojde opět k uvolnění paměti a z rozdílu hodnot aktivního a neaktivního streamu s kodekem H.265 vychází celková spotřeba paměti na 0,32 GB. Využití paměti je tedy srovnatelné s kodekem H.264.



Obrázek 5.22: Využití RAM s aktivním streamem (MPEG-4 Video) 2000 kbit/s

Poslední kodek MPEG-4 Video je pro totožné rychlosti vyzkoušen společně s kodekem MP3 a celkové využití paměti serveru je 1,72 GB.



Obrázek 5.23: Využití RAM s neaktivním streamem (MPEG-4 Video) 2000 kbit/s

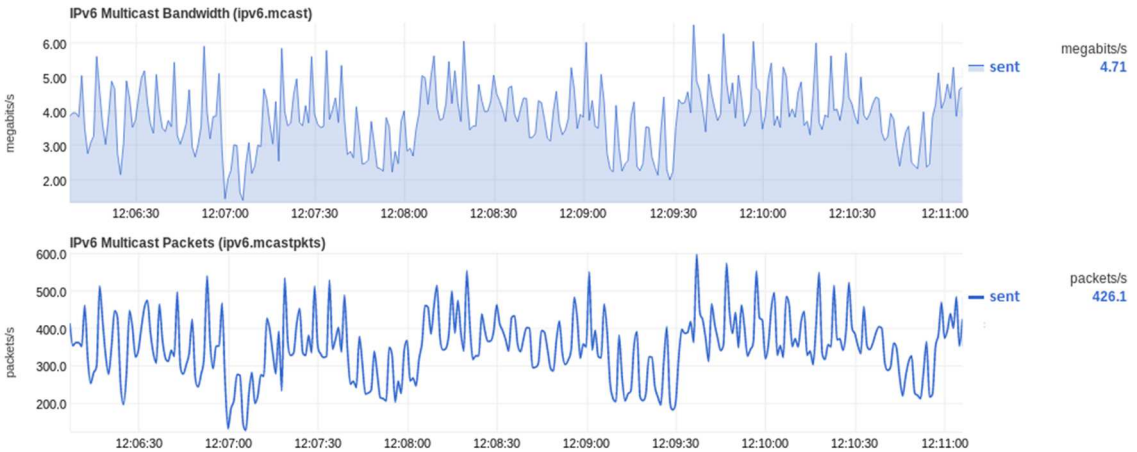
Ve vztahu ke konkurenčním kodekům H.264 a H.265 je tento druh k operační paměti velice šetrný. Zlom z obrázku 5.22 není příliš viditelný, ale výpočtem celkového využití paměti pro stream vychází hodnota na 0,05 GB.

5.5.6 Využití rozhraní serveru pro kodeky H.265, H.264 a MPEG-4 Video

Aktivní stream server s rozdílnými kodeky nemusí souviset pouze s využitím systémových prostředků, ale také s využitím rozhraní. Kodeky s odlišnými parametry mohou vést k různým nárokům na rozhraní stream serveru, čímž dochází k ovlivnění jejich parametrů. S pomocí programu netdata je sledován parametr využití šířky přenosového pásma a počet odeslaných paketů pro dvě přenosové rychlosti. Video kodeky H.265, H.264 a MPEG-4 Video jsou stejně jako u systémových prostředků doplněny o kodek MP3 (320 kbit/s) pro audio data.

U kodeku H.264 s rychlostí 2000 kbit/s je využití šířky pásma velice proměnlivé. Hodnoty tohoto parametru při konstantně spuštěném streamu dosahují 1 až 4 Mbit/s s možností překročení horní hranice. Graf odpovídající počtu odeslaných paketů kopíruje strukturu grafu šířky pásma a s počtem těchto paketů souvisí i míra využití pásma.

Obrázek 5.24: Využití šířky pásma a počet odeslaných paketů - H.264 (2000 kbit/s)



Obrázek 5.25: Využití šířky pásma a počet odeslaných paketů - H.264 (10 000 kbit/s)

Pro 10 000 kbit/s jsou hranice využití šířky pásma a počet odesalných paketů poměrně vyšší. S pětinásobným nárustem přenosové rychlosti jsou hodnoty využití pásma v rozmezí 2 až 6,5 Mbit/s a z výsledů lze usoudit, že komprimace kodeku H.264 pro vyšší přenosové rychlosti prokazuje dobrou účinnost.

Kodek H.265 je nástupcem staršího kodeku H.264. Výhodou tohoto kodeku, je kromě podpory technologie UHD i efektnivnější komprese, a tím dosahuje i snížení hodnoty využití šířky pásma. Efektivní komprese se projevuje již pro nižší přenosovou rychlost 2000 kbit/s, kde hodnoty využití pásma se pohybují výrazně níže, než u kodeku H.264 pro stejnou rychlost.

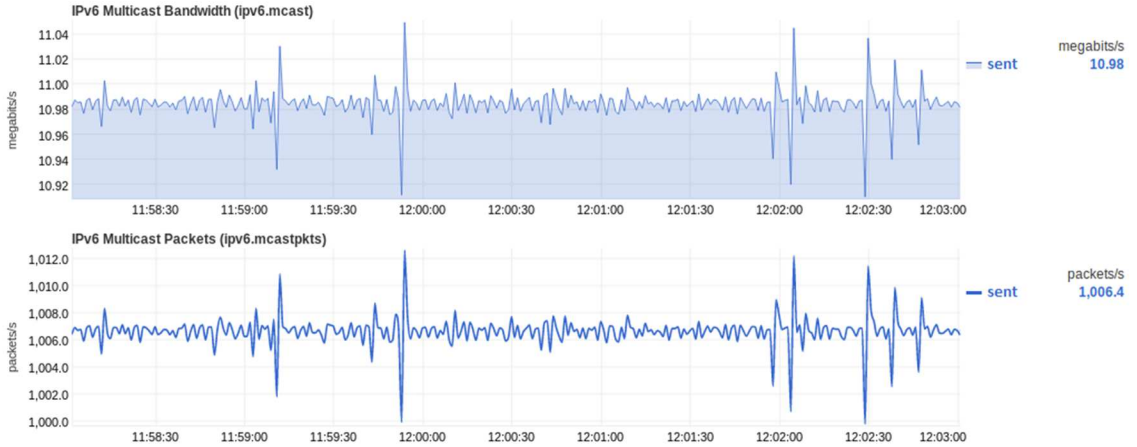
Obrázek 5.26: Využití šířky pásma a počet odeslaných paketů - H.265 (2000 kbit/s)

Obrázek 5.27: Využití šířky pásma a počet odeslaných paketů - H.265 (10 000 kbit/s)

V porovnání vyšší přenosové rychlosti 10 000 kbit/s H.264 s novějším H.265 není využití šířky pásma příliš rozdílné. Hodnoty se pohybují v rozmezí 2 až 7 Mbit/s s možností překročení horní hranice až na 8 Mbit/s. Z těchto výsledků je zřejmé, že efektnivnější komprese kodeku se projevuje spíše pro nižší přenosové rychlosti.

U posledního typu kodeku MPEG-4 Video pro obě přenosové rychlosti jsou průběhy grafů podstatně konstantnější než u předchozích dvou typů. Nicméně komprese MPEG-4 Video není příliš kvalitní a to ani při zvýšení přenosové rychlosti.

Obrázek 5.28: Využití šířky pásma a počet odeslaných paketů - MPEG-4 Video (2000 kbit/s)



Obrázek 5.29: Využití šířky pásma a počet odeslaných paketů - MPEG-4 Video (10 000 kbit/s)

Závěr

Cílem diplomové práce je navržení otevřeného řešení pro streamování videí pomocí multicastu v prostředí IPv6 sítí, založené na open source routovacích platformách a programech. Součástí řešení je studium a popis multicastu včetně popisu kodeků sloužících pro zpracování audio a video dat. Společně s kodeky a multicastem jsou součástí práce také konfigurace linuxový programů, konfigurace linuxových zařízení pomocí bash skriptů a zátěžové testování.

Veškeré testování se uskutečnilo na vlastních síťových topologiích za přítomnosti zdroje multicastu, směrovačů a několika příjemců. Zdroj společně s příjemci využívají program VLC Media Player, který slouží jako stream server a také jako možnost příjmu pro uživatele.

Pro navržení otevřeného řešení pomocí open source nástrojů a testování v laboratoři je nutné zvolit nástroje pro vytvoření směrovačů v síťové topologii, na kterých budou testovány. Jedním z řešení je možnost využití platformy Xorp. Platforma Xorp dokáže pracovat s multicastem jak v síti IPv4, tak IPv6, ovšem zprovoznění multicastového přenosu je dosaženo pouze u IPv4. V přenosu realizovaného na IPv6 síti je odhalena zásadní chyba ve funkčnosti programu Xorp, konkrétně u protokolu PIM. Tento protokol je základní součástí multicastového přenosu pro distribuci dat sítí a problém je neschopnost generovat zprávu Register. Pokud zdroj multicastu vysílá data, měl by první směrovač v síti všechny data přeposílat směrem k RP. Na počátku vysílání první směrovač neposílá všechny data, ale pouze první paket, který zapouzdří do zprávy Register a tato zpráva je poslána RP směrovači. Po obdržení zprávy je RP informován o přítomnosti zdroje s danou multicastovou adresou a s příchodem zprávy Join od směrovačů s připojeným příjemcem by měl RP distribuovat data skrze síť. Zpráva Register není v komunikaci přítomna a RP o zdroji multicastu nemá žádné informace. Z pohledu dalšího vývoje testování by bylo vhodné nalézt chybu v programu Xorp 1.8.6, nicméně pro vysílání multicastových dat v IPv6 sítí není tento program příliš vhodný.

Po neúspěšném testování programu Xorp v IPv6 bylo potřeba otestovat nástroj MRD6. Program MRD6 je velmi jednoduchý a volně dostupný z balíčkového systému Linux. Podobně jako u programu Xorp i zde nedospělo testování k úspěšnému přenosu. Po veškeré instalaci, kontrole konfiguračních souborů a výpisů, které jsou v pořádku je zjištěna chyba opětovně v samotném programu. MRD6 nedokáže v systému povolit parametr mc_forwarding, který je důležitý pro přesměrování multicastu. Vzhledem k této skutečnosti, byly vyzkoušeny kroky k manuálnímu povolení parametru, ovšem bez úspěchu. Použití programu není tedy příliš vhodné, ale nalezením řešení pro povolení přesměrování by se jednalo o jednoduchý a velmi efektivní nástroj.

Závěrečný nástroj pro realizaci multicastového přenosu je program SMCRoute. Podobně jako MRD6 je i tento program dostupný z balíčkového systému a jeho použití je velmi jednoduché. Po instalaci a nastavení všech konfiguračních souborů dospělo testování úspěšného přenosu streamu skrze počítačovou síť. Díky malé velikosti souboru a snadné práci se jedná o velmi efektivní nástroj pro přenos multicastových dat. Vzhledem ke správné funkcionalitě je měření obohaceno o práci s kodeky a konkrétně jejich vlivem na vytížení procesoru, operační

paměti RAM a rozhraní stream serveru. Pro práci s procesorem a operační pamětí jsou vyzkoušeny tři kodeky H.265, H.264 a MPEG-4 Video, které dnes patří mezi nejznámější a také nejvyužívanější v oblasti streamu a televizního vysílání. Všechny video kodeky jsou testovány s totožnou přenosovou rychlostí 2000, 10 000 a 20 000 kbit/s společně s kodekem MP3, který je určen pro kompresi audio dat s rychlostí 320 kbit/s. Z výsledků testování vychází najevo, že nejmenší dopad na výkon procesoru má kodek H.264 a lze jej společně s nástrojem SMCRoute považovat za ideální variantu pro streamování. Kodek, který lze podle výsledků klasifikovat jako nevhodný je H.265. Jeho vytížení procesoru je velmi vysoké. U operační paměti RAM jsou výsledky kodeků H.264 a H.265 totožné, ale zdaleka nejnižší dopad na paměť má kodek MPEG-4 Video. Hodnoty využití paměti kodeků H.264 a H.265, i přes nižší hodnotu kodeku MPEG-4 Video, nejsou příliš vysoké a je možné s těmito kodeky pracovat. Pro práci s rozhraními stream serveru jsou využity opět kodeky H.265, H.264 a MPEG-4 Video, ale pouze pro dvě přenosové rychlosti 2000 a 10 000 kbit/s, kde je sledováno využití šířky pásma a počet přenesených paketů. Z výsledků testování je kodek H.265 považován za nejefektivnější, naopak nejméně efektivním kodekem je MPEG-4 Video.

Po vyhodnocení všech testovaných programů lze konstatovat, že nejlepším a také funkčním nástrojem pro streamování videí pomocí multicastu v IPv6 sítích je program SMCRoute společně s kodekem H.264. Z pohledu vytížení procesoru dosahuje tento kodek nejnižších hodnot a v případě využití operační paměti je srovnatelný s kodekem H.265, jehož hodnoty nejsou vysoké. Co se týče rozhraní, pak konkurenční kodek H.265 je z pohledu využití šířky pásma lepší, ale tato výhoda je vykoupena příliš velným využitím procesoru, čímž je ovlivněna i stabilita streamu a není vhodné tento kodek používat. Všechny testované programy jsou testovány na zařízení v laboratoři společně s Gigabit Ethernetovými rozhraními a Fast Ethernetovými linkami. Jedná se tak o lepší variantu k sestavení sítě než v roce 2017, kdy se má bakalářská práce zabývala podobným tématem. V bakalářské práci z roku 2017 jsou k vytvoření síťových topologií použity sériové linky na směrovačích Cisco a rychlost těchto linek není možné porovnávat s připojením Fast Ethernet. Díky využití rychlejších přenosových médií, jiného prostředí a práci na jiných zařízení než Cisco, bylo možné prohloubit své znalosti a obohatit se o zkušenosti s multicastem v systému Linux.

Použitá literatura

[1] SATRAPA, Pavel. IPv6: Internetový protokol verze 6. 3., aktualiz. a dopl. vyd. Praha: CZ.NIC, c2011. ISBN 978-80-904248-4-5.

[2] MINOLI, Daniel. Linear and Non-Linear Video and TV Applications: Using IPv6 and IPv6 Multicast. Canada: Wiley, c2012. ISBN 978-1-118-18658-9.

[3] HORÁK, Jaroslav a Milan KERŠLÁGER. Počítačové sítě pro začínající správce. 3. aktualizované vydání. Brno: Computer Press, 2006. ISBN 80-251-0892-9.

[4] SZE, Vivienne, Madhukar BUDAGAVI a Gary SULLIVAN. High Efficiency Video Coding (HEVC): Algorithms and Architectures. Švýcarsko: Springer, 2014. ISBN 978-3-319-06894-7.

[5] MER, Přemysl. Multimediální technika pro integrovanou výuku VUT a VŠB-TUO [elektronická skripta]. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2014 [cit. 2019-03-12]. ISBN 978-80-248-3568-6. Dostupné z: http://9.9e.cz/mut.pdf

[6] NEVLUD, Pavel. Komunikační sítě II pro integrovanou výuku VUT a VŠB-TUO [elektronická skripta]. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2014 [cit. 2019-03-12]. ISBN 978-80-248-3639-3. Dostupné z: https://docplayer.cz/18160574-Komunikacni-site-ii-pro-integrovanou-vyuku-vut-a-vsb-tuo.html

[7] Xorp User Manual: Version 1.8-CT. In: Xorp [online]. London, 2010 [cit. 2019-03-12]. Dostupné z: http://www.xorp.org/releases/1.8-CT/docs/user_manual/user_manual.pdf

[8] BOUŠKA, Petr. Počítačové sítě a jejich typy. Samuraj-cz [online]. Petr Bouška, c2005-2019, 09.07.2007 [cit. 2019-03-12]. Dostupné z: https://www.samuraj-cz.com/clanek/pocitacove-site-a-jejich-typy/

[9] Computer network. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/Computer_network

[10] Počítačové sítě. Střední škola technická Opava [online]. c2011 [cit. 2019-03-12]. Dostupné z: https://sst.opava.cz/ict/site/ps_rozdeleni.pdf

[11] BOUŠKA, Petr. TCP/IP - skupinové vysílání IP Multicast a Cisco. Samuraj-cz [online]. Petr Bouška, c2005-2019, 10.03.2009 [cit. 2019-03-12]. Dostupné z: https://www.samuraj-cz.com/clanek/tcpip-skupinove-vysilani-ip-multicast-a-cisco/

[12] Multicast v CCTV. Abbas [online]. Brno: ABBAS, c2011-2019, 14.12.2015 [cit. 2019-03-12]. Dostupné z: http://www.abbas.cz/clanky/recenze-technika/multicast-v-cctv/

[13] Multicast. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/Multicast

[14] Unicast. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/Unicast

[15] Broadcasting (networking). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/Broadcasting_(networking)

[16] Anycast. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/Anycast

[17] How To Configure Multicast Paging. Snom [online]. [cit. 2019-03-12]. Dostupné z: https://helpdesk.snom.com/support/solutions/articles/6000194132-how-to-configure-multicast-paging

[18] What Is a Codec and Why Do I Need It. Lifewire [online]. [cit. 2019-03-12]. Dostupné z: https://www.lifewire.com/what-exactly-is-odec-2483426

[19] Codec. VideoLAN Wiki [online]. [cit. 2019-03-12]. Dostupné z: https://wiki.videolan.org/Codec/

[20] MPEG - Moving Picture Experts Group. Webopedia [online]. c2019 [cit. 2019-03-12]. Dostupné z: https://www.webopedia.com/TERM/M/MPEG.html

[21] MPEG-1. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/MPEG-1

[22] MPEG-2. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/MPEG-2

[23] MPEG-3. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/MPEG-3

[24] MPEG-4. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/MPEG-4

[25] What is H.264. H264info [online]. c2007-2010 [cit. 2019-03-12]. Dostupné z: http://www.h264info.com/h264.html

[26] H.264/MPEG-4 AVC. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

[27] H.265/HEVC - kodek, ktorý bude vládnuť svetu?. TECHBOX [online]. c2010-2019 [cit. 2019-03-12]. Dostupné z: https://techbox.dennikn.sk/temy/h-265-hevc-kodek-ktory-bude-vladnut-svetu/

[28] Vše o vysílání v DVB-T2 / H.265 HEVC v ČR. ANTÉNA [online]. Malý, c2003-2016 [cit. 2019-03-12]. Dostupné z: https://www.antena.cz/vse-o-vysilani-v-dvb-t2-h-265-hevc-v-cr-c216/

[29] High Efficiency Video Coding. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/High_Efficiency_Video_Coding

[30] FFmpeg. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/FFmpeg

[31] About FFmpeg. FFmpeg [online]. [cit. 2019-03-12]. Dostupné z: https://www.ffmpeg.org/about.html

[32] Rozdíl mezi FLAC a MP3. Kdy jaký formát použít a proč?. Letem světem Applem [online]. c2011-2019 [cit. 2019-03-12]. Dostupné z: https://www.letemsvetemapplem.eu/2015/02/22/rozdil-flac-mp3-jaky-format-pouzit-proc/

[33] What is FLAC? Xiph [online]. Coalson, c2000-2009 [cit. 2017-03-21]. Dostupné z: https://xiph.org/flac/index.html

[34] FLAC. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/FLAC

[35] Vorbis. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/Vorbis

[36] Vorbis audio compression. Xiph [online]. c1994-2016 [cit. 2019-03-12]. Dostupné z: https://xiph.org/vorbis/

[37] Vorbis I specification. Xiph [online]. c1994-2016 [cit. 2019-03-12]. Dostupné z: https://xiph.org/vorbis/doc/Vorbis_I_spec.html

[38] Formát AAC. Programujte [online]. Webtea, c2003-2019 [cit. 2019-03-12]. Dostupné z: http://programujte.com/clanek/2006050705-format-aac/

[39] AC3 File Format. WhatIs [online]. TechTarget, c1999-2019 [cit. 2019-03-12]. Dostupné z: https://whatis.techtarget.com/fileformat/AC3-Dolby-Digital-audio-files-used-on-DVDs

[40] Dolby Digital. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/Dolby_Digital

[41] MP3. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/MP3

[42] MP3. Computer Hope [online]. c2019 [cit. 2019-03-12]. Dostupné z: https://www.computerhope.com/jargon/m/mp3.htm

[43] What Is An MP3 File (And How Do I Open One)?. How-To Geek [online]. c2019 [cit. 2019-03-12]. Dostupné z: https://www.howtogeek.com/361516/what-is-an-mp3-file-and-how-do-i-open-one/

[44] Apple Lossless. Apple Lossless [online]. c2018 [cit. 2019-03-12]. Dostupné z: http://www.applelossless.com/

[45] Apple Lossless. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-12]. Dostupné z: https://en.wikipedia.org/wiki/Apple_Lossless

[46] Co je WMA?. Apowersoft [online]. c2019 [cit. 2019-03-12]. Dostupné z: https://www.apowersoft.cz/co-je-wma

[47] Co je to WMV formát?. Apowersoft [online]. c2019 [cit. 2019-03-12]. Dostupné z: https://www.apowersoft.cz/co-je-to-wmv-format

[48] Welcome to XORP. Xorp [online]. c2008-2010 [cit. 2019-03-12]. Dostupné z: http://www.xorp.org/

[49] XORP Documentation Project. Xorp [online]. c2008-2010 [cit. 2019-03-12]. Dostupné z: http://xorp.run.montefiore.ulg.ac.be/start

[50] Introduction. Xorp [online]. c2008-2010 [cit. 2019-03-12]. Dostupné z: http://xorp.run.montefiore.ulg.ac.be/latex2wiki/getting_started

[51] What's LXD?. Linux Containers [online]. [cit. 2019-03-12]. Dostupné z: https://linuxcontainers.org/lxd/introduction/

[52] Infrastructure for container projects. Linux Containers [online]. [cit. 2019-03-12]. Dostupné z: https://linuxcontainers.org/

[53] Virtualizace v Linuxu. Wikiknihy [online]. 2017 [cit. 2019-03-12]. Dostupné z: https://cs.wikibooks.org/wiki/Virtualizace_v_Linuxu

[54] LXD (Linux container hypervisor). TechTarget [online]. TechTarget, c2016-2019 [cit. 2019-03-12]. Dostupné z: https://searchitoperations.techtarget.com/definition/LXD-Linux-container-hypervisor

[55] Linuxové kontejnery. TURRIS [online]. [cit. 2019-03-12]. Dostupné z: https://doc.turris.cz/doc/cs/howto/lxc

[56] What's LXCFS?. Linux Containers [online]. [cit. 2019-03-13]. Dostupné z: https://linuxcontainers.org/lxcfs/introduction/

[57] What's LXC?. Linux Containers [online]. [cit. 2019-03-13]. Dostupné z: https://linuxcontainers.org/lxc/introduction/

[58] Install LXD pure-container hypervisor on Ubuntu 18.04 LTS. NixCraft [online]. c2000-2019 [cit. 2019-03-13]. Dostupné z: https://www.cyberciti.biz/faq/install-lxd-pure-container-hypervisor-on-ubuntu-18-04-lts/

[59] Clustering. LXD - system container manager [online]. [cit. 2019-03-13]. Dostupné z: https://lxd.readthedocs.io/en/latest/clustering/

[60] VLC media player. VideoLAN Wiki [online]. [cit. 2019-03-13]. Dostupné z: https://wiki.videolan.org/VLC_media_player/

[61] VideoLAN. VideoLAN Wiki [online]. [cit. 2019-03-13]. Dostupné z: https://wiki.videolan.org/VideoLAN/

[62] VLC media player. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-13]. Dostupné z: https://en.wikipedia.org/wiki/VLC_media_player

[63] Documentation:Command line. VideoLAN Wiki [online]. [cit. 2019-03-13]. Dostupné z: https://wiki.videolan.org/Documentation:Command_line/

[64] Streaming Features list: VLC Stream output. VideoLAN [online]. [cit. 2019-03-13]. Dostupné z: https://www.videolan.org/streaming-features.html

[65] Documentation:Streaming HowTo/Advaced Streaming Using the Command Line. VideoLAN Wiki [online]. [cit. 2019-03-13]. Dostupné z: https://wiki.videolan.org/Documentation:Streaming_HowTo/Advanced_Streaming_Using_the_Command_Line/

[66] Documentation:Streaming HowTo/Advaced Streaming over IPv6. VideoLAN Wiki [online]. [cit. 2019-03-13]. Dostupné z: https://wiki.videolan.org/Documentation:Streaming_HowTo/Streaming_over_IPv6/

[67] VLC Media Player 3.0.4. Neowin [online]. c2000 [cit. 2019-03-13]. Dostupné z: https://www.neowin.net/news/vlc-media-player-304/

[68] VLC 3.0.4 Vetinari. VideoLAN [online]. [cit. 2019-03-13]. Dostupné z: https://www.videolan.org/vlc/releases/3.0.4.html

[69] LXD - new interface Ubuntu 18.04. Linux Containers Forum [online]. [cit. 2019-03-13]. Dostupné z: https://discuss.linuxcontainers.org/t/lxd-new-interface-ubuntu-18-04/3234

[70] Documentation:Streaming HowTo/Command Line Examples. VideoLAN Wiki [online]. [cit. 2019-03-13]. Dostupné z: https://wiki.videolan.org/Documentation:Streaming_HowTo/Command_Line_Examples/

[71] Static Multicast Routing Daemon. Troglobit [online]. Nilson, 2019 [cit. 2019-03-13]. Dostupné z: http://troglobit.com/projects/smcroute/

[72] SMCRoute. Manned [online]. [cit. 2019-03-13]. Dostupné z: https://manned.org/smcroute/ee34e68d

[73] IPv6 Multicast Routing Daemon. Debian [online]. c1997-2018 [cit. 2019-03-13]. Dostupné z: https://packages.debian.org/stable/net/mrd6

[74] MRD6. Linux Certif [online]. c2006-2019 [cit. 2019-03-13]. Dostupné z: http://www.linuxcertif.com/man/8/mrd6/

[75] Multicast Routing Daemon v6. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-13]. Dostupné z: https://en.wikipedia.org/wiki/Multicast_Routing_Daemon_v6

[76] Compiled language. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-13]. Dostupné z: https://en.wikipedia.org/wiki/Compiled_language

[77] Úvod do skriptování v Linuxu. Root.cz [online]. Internet Info, c1998-2019 [cit. 2019-03-13]. Dostupné z: https://www.root.cz/clanky/uvod-do-skriptovani-v-linuxu/

[78] Scripting language. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-13]. Dostupné z: https://en.wikipedia.org/wiki/Scripting_language

[79] BASH: Skriptování. Milan Kerslager [online]. Kerslager [cit. 2019-03-13]. Dostupné z: https://www.pslib.cz/milan.kerslager/BASH:_Skriptov%C3%A1n%C3%AD

[80] Programování v (bash) shellu. Root.cz [online]. Internet Info, c1998-2019 [cit. 2019-03-13]. Dostupné z: https://www.root.cz/clanky/programovani-v-bash-shellu/

[81] Bash pokračování. Ubuntu Česká republika [online]. [cit. 2019-03-13]. Dostupné z: https://wiki.ubuntu.cz/bash_pokra%C4%8Dov%C3%A1n%C3%AD

Seznam příloh

Příloha A: *Bash skripty pro testování programu Xorp v IPv6*

Bash skript pro stream server

Bash skript pro stream server, příjemce a směrovače v IPv6 jsou totožné se skripty, které slouží v IPv4. Nicméně pro server a příjemce v IPv4 je potřeba upravit adresování.

```
#!/bin/bash
apt-get update
apt-get install -y vlc
echo Nastavuji IPv6 adresu stroje a vychozi branu.
sleep 3
service network-manager stop
ip link set enp1s0 down
ip -6 route del default
ip addr add 2001:db8:ab:100::10/64 dev nazevRozhrani
ip -6 route add default via 2001:db8:ab:100::1
ip link set nazevRozhrani up
((count = 10))
while [[ $count -ne 0 ]] ; do
    ping -c 3 2001:db8:ab:100::1
    rc=$?
    if [[ $rc -eq 0 ]] ; then
        ((count = 1))
    fi
    ((count = count - 1))
done
if [[ $rc -eq 0 ]] ; then
    echo "Rozhrani je pripojeno."
else
    echo "Rozhrani neni pripojeno."
Fi
```

Bash skript pro příjemce multicastu

```
#!/bin/bash
apt-get update
apt-get install -y vlc

echo Nastavuji IPv6 adresu stroje a vychozi branu.
sleep 3
service network-manager stop
ip link set enp1s0 down
ip -6 route del default
ip addr add IPv6_adresa/64 dev nazevRozhrani
ip -6 route add default via IPv6_adresa
ip link set nazevRozhrani up
((count = 10))
while [[ $count -ne 0 ]] ; do
    ping -c 3 IPv6_adresa
    rc=$?
    if [[ $rc -eq 0 ]] ; then
        ((count = 1))
    fi
    ((count = count - 1))
done
if [[ $rc -eq 0 ]] ; then
    echo "Rozhrani je pripojeno."
else
    echo "Rozhrani neni pripojeno."
Fi
```

Bash skript pro směrovače

```
#!/bin/bash
echo Instaluji nastroje pro spravu programu Xorp.
sleep 3
apt-get install -y unzip nano tcpdump
unzip /home/ubuntu/xorp.ct-master.zip -d /home/ubuntu/
mv /home/ubuntu/xorp.ct-master /home/ubuntu/xorp.ct

echo Instaluji potrebne balicky.
sleep 3
apt-get install -y build-essential git scons libboost-all-dev
libssl-dev libncurses5-dev libpcap-dev traceroute flex bison
cd /home/ubuntu/xorp.ct/xorp
echo Probehne kompilace a instalace programu Xorp.
scons
scons install
if [ ! -e /usr/local/xorp/sbin/xorp_rtrmgr ]; then
    echo "Instalace neprobehla spravne."
else
    echo "Kompilace a instalace probehla spravne."
fi
echo Kopiruji konfiguracni soubor smerovace.
sleep 3
cp /home/ubuntu/soubor.boot /usr/local/xorp/sbin/
cd /usr/local/xorp/sbin/
groupadd xorp
usermod -a -G xorp root
echo Spoustim smerovac.
sleep 3
/usr/local/xorp/sbin/xorp_rtrmgr -b
/usr/local/xorp/sbin/soubor.boot
```

Příloha B: *Konfigurace směrovačů R1 - R3 programu Xorp pro IPv6*

V této sekci jsou uvedeny konfigurační soubory směrovačů R1 až R3. Zbylé konfigurační soubory pro směrovače R4 a R5 jsou totožné s konfigurací směrovače R3 ovšem s odlišným adresováním dle topologie sítě.

Konfigurace směrovače R1 (Xorp)

```
/* $XORP$ */
interfaces {
    interface eth1 {
    disable: false
        vif eth1 {
     disable: false
            address 2001:db8:ab:100::1 {
                prefix-length: 64
         disable: false
            }
            address fe80::1111 {
                prefix-length: 64
         disable: false
            }
        }
    }
    interface eth2 {
    disable: false
        vif eth2 {
            address 2001:db8:ab:200::1 {
                prefix-length: 64
         disable: false
            }
            address fe80::2222 {
                prefix-length: 64
         disable: false
            }
        }
```

```
    }
}
fea {
    unicast-forwarding6 {
        disable: false
    }
}
plumbing {
    mfea6 {
        disable: false
        interface eth1 {
            vif eth1 {
                disable: false
            }
        }
        interface eth2 {
            vif eth2 {
                disable: false
            }
        }
        interface register_vif {
            vif register_vif {
                disable: false
            }
        }
     traceoptions {
           flag all {
                  disable: false
           }
     }
      }
}
protocols {
```

```
ospf6 0 {
    router-id: 10.10.10.10
    area 0.0.0.0 {
        interface eth1 {
            vif eth1 {
            }
        }
        interface eth2 {
            vif eth2 {
                }
            }
        }
}
mld {
    disable:false
    interface eth1 {
        vif eth1 {
            disable: false
            version: 2
        }
    }
    interface eth2 {
        vif eth2 {
            disable: false
            version: 2
        }
    }
 traceoptions {
 flag all {
      disable:false
 }
 }
}
```

```
    pimsm6 {
        disable:false
        interface eth1 {
            vif eth1 {
                disable: false
            }
        }
        interface eth2 {
            vif eth2 {
                disable: false
            }
        }
        interface register_vif {
           vif register_vif {
           disable: false
           }
     }
        static-rps {
            rp 2001:db8:ab:300::2 {
                group-prefix ff00::/8 {
                }
            }
        }
     traceoptions {
           flag all {
                disable: false
           }
           }
     }
    fib2mrib {
        disable: false
    }
}
```

Konfigurace směrovače R2 (Xorp)

```
/* $XORP$ */
interfaces {
    interface eth1 {
        disable: false
        vif eth1 {
     disable: false
            address 2001:db8:ab:200::2 {
                prefix-length: 64
         disable: false
            }
            address fe80::3333 {
                prefix-length: 64
         disable: false
            }
        }
    }
    interface eth2 {
        disable: false
        vif eth2 {
     disable: false
            address 2001:db8:ab:300::1 {
                prefix-length: 64
         disable: false
            }
            address fe80::4444 {
                prefix-length: 64
         disable: false
            }
        }
    }
}
```

```
fea {
    unicast-forwarding6 {
        disable: false
    }
}
plumbing {
    mfea6 {
        disable: false
        interface eth1 {
            vif eth1 {
                disable: false
            }
        }
        interface eth2 {
            vif eth2 {
                disable: false
            }
        }
        interface register_vif {
            vif register_vif {
                disable: false
            }
        }
   traceoptions {
        flag all {
            disable: false
        }
    }
    }
}
protocols {
    ospf6 0 {
        router-id: 20.20.20.20
```

```
        area 0.0.0.0 {
            interface eth1 {
                vif eth1 {
                }
            }
            interface eth2 {
                vif eth2 {
                    }
                }
        }
    }
    mld {
        disable:false
        interface eth1 {
            vif eth1 {
                disable: false
                version: 2
            }
        }
        interface eth2 {
            vif eth2 {
                disable: false
                version: 2
            }
        }
        traceoptions {
           flag all {
                disable:false
            }
      }
    }
    pimsm6 {
       disable:false
```

```
        interface eth1 {
            vif eth1 {
                disable: false
            }
        }
        interface eth2 {
            vif eth2 {
                disable: false
            }
        }
        interface register_vif {
        vif register_vif {
        disable: false
        }
    }
        static-rps {
            rp 2001:db8:ab:300::2 {
                group-prefix ff00::/8 {
                }
            }
        }

  traceoptions {
        flag all {
            disable: false
        }
      }
   }

    fib2mrib {
        disable: false
    }
}
```

Konfigurace směrovače R3 (Xorp)

```
/* $XORP$ */
interfaces {
    interface eth1 {
        disable: false
        vif eth1 {
     disable: false
            address 2001:db8:ab:300::2 {
                prefix-length: 64
                 disable: false
            }
            address fe80::5555 {
                prefix-length: 64
                 disable: false
            }
        }
    }
    interface eth2 {
        disable: false
        vif eth2 {
     disable: false
            address 2001:db8:ab:400::1 {
                prefix-length: 64
         disable: false
            }
            address fe80::6666 {
                prefix-length: 64
         disable: false
            }
        }
    }
    interface eth3 {
        disable: false
```

```
        vif eth3 {
     disable: false
            address 2001:db8:ab:600::1 {
                prefix-length: 64
         disable: false
            }
            address fe80::7777 {
                prefix-length: 64
         disable: false
            }
        }
    }
}
fea {
    unicast-forwarding6 {
        disable: false
    }
}
plumbing {
    mfea6 {
        disable: false
        interface eth1 {
            vif eth1 {
                disable: false
            }
        }
        interface eth2 {
            vif eth2 {
                disable: false
            }
        }
       interface eth3 {
            vif eth3 {
```

```
                disable: false
            }
        }
        interface register_vif {
            vif register_vif {
                disable: false
            }
    }
    traceoptions {
        flag all {
            disable: false
        }
      }
    }
}
protocols {
    ospf6 0 {
        router-id: 30.30.30.30
        area 0.0.0.0 {
            interface eth1 {
                vif eth1 {
                }
            }
            interface eth2 {
                vif eth2 {
                    }
                }
            interface eth3 {
                vif eth3 {
                    }
                }
        }
    }
```

```
mld {
    disable:false
    interface eth1 {
        vif eth1 {
            disable: false
            version: 2
        }
    }
    interface eth2 {
        vif eth2 {
            disable: false
            version: 2
        }
    }
    interface eth3 {
        vif eth3 {
            disable: false
            version: 2
        }
    }
traceoptions {
flag all {
    disable:false
   }
  }
}
pimsm6 {
    disable:false
    interface eth1 {
        vif eth1 {
            disable: false
        }
    }
```

```
        interface eth2 {
            vif eth2 {
                disable: false
            }
        }
       interface eth3 {
            vif eth3 {
                disable: false
            }
        }
        interface register_vif {
        vif register_vif {
        disable: false
        }
    }
        static-rps {
            rp 2001:db8:ab:300::2 {
                group-prefix ff00::/8 {
                }
            }
        }

    traceoptions {
        flag all {
            disable: false
        }
      }
    }
    fib2mrib {
        disable: false
    }
}
```

Příloha C: *Výpisy z testování programu Xorp pro IPv6*

Výpisy směrovače R1 (Xorp)

```
root@router-xorp:~# sysctl -a | grep ipv6 | grep forward
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.all.mc_forwarding = 1
net.ipv6.conf.default.forwarding = 1
net.ipv6.conf.default.mc_forwarding = 0
net.ipv6.conf.eth0.forwarding = 1
net.ipv6.conf.eth0.mc_forwarding = 0
net.ipv6.conf.eth1.forwarding = 1
net.ipv6.conf.eth1.mc_forwarding = 1
net.ipv6.conf.eth2.forwarding = 1
net.ipv6.conf.eth2.mc_forwarding = 1
net.ipv6.conf.lo.forwarding = 1
net.ipv6.conf.lo.mc_forwarding = 0
net.ipv6.conf.pim6reg253.forwarding = 1
net.ipv6.conf.pim6reg253.mc_forwarding = 1


root@router-xorp:/usr/local/xorp/sbin# tcpdump -i eth1
13:44:47.370112 IP6 2001:db8:ab:100::10.54195 >
ff7e:240:2001:db8:ab:300:0:20.1234: UDP, length 1316
13:44:47.371004 IP6 2001:db8:ab:100::10.54195 >
ff7e:240:2001:db8:ab:300:0:20.1234: UDP, length 1316
13:44:47.371005 IP6 2001:db8:ab:100::10.54195 >
ff7e:240:2001:db8:ab:300:0:20.1234: UDP, length 1316


root@router-xorp:/usr/local/xorp/sbin# tcpdump -i eth2
13:46:14.891664 IP6 fe80::3333 > ff02::5: OSPFv3, Hello, length
40
13:46:22.591819 IP6 fe80::2222 > ff02::d: PIMv2, Hello, length
56
13:46:23.014573 IP6 fe80::2222 > ff02::5: OSPFv3, Hello, length
40
13:46:24.891525 IP6 fe80::3333 > ff02::5: OSPFv3, Hello, length
40
```

```
13:46:25.482197 IP6 fe80::3333 > ff02::d: PIMv2, Hello, length


root@router-xorp:/usr/local/xorp/sbin# tcpdump -i eth1
14:01:38.188607 IP6 2001:db8:ab:100::10.35264 >
ff7e:240:2001:db8:ab:300:0:20.1234: UDP, length 940
14:01:38.188969 IP6 2001:db8:ab:100::10.35264 >
ff7e:240:2001:db8:ab:300:0:20.1234: UDP, length 940
14:01:38.189343 IP6 2001:db8:ab:100::10.35264 >
ff7e:240:2001:db8:ab:300:0:20.1234: UDP, length 940


root@router-xorp> show interfaces eth1
eth1/eth1: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet6 2001:db8:ab:100::1 prefixlen 64
        inet6 fe80::1111 prefixlen 64
        physical index 6
        ether 0:13:3b:a0:4:2a


root@router-xorp> show interfaces eth2
eth2/eth2: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet6 2001:db8:ab:200::1 prefixlen 64
        inet6 fe80::2222 prefixlen 64
        physical index 7
        ether 0:13:3b:9c:bf:3f


root@router-xorp> show mfea6 interface
Interface    State    Vif/PifIndex Addr            Flags
eth1         UP               0/6 2001:db8:ab:100::1 MULTICAST
BROADCAST KERN_UP
eth2         UP               1/7 2001:db8:ab:200::1 MULTICAST
BROADCAST KERN_UP
register_vif UP               2/6 2001:db8:ab:100::1
PIM_REGISTER KERN_UP
```

```
root@router-xorp> show mld interface
Interface    State    Querier         Timeout Version Groups
eth1         UP       fe80::1111         None       2     10
eth2         UP       fe80::2222         None       2     13

root@router-xorp> show pim6 interface
Interface    State    Mode   V PIMstate Priority DRaddr
Neighbors
eth1         UP       Sparse 2 DR              1 fe80::1111
0
eth2         UP       Sparse 2 NotDR           1 fe80::3333
1
register_vif UP       Sparse 2 DR              1 fe80::1111
0

root@router-xorp> show pim6 mrib
DestPrefix         NextHopRouter   VifName VifIndex MetricPref
Metric
2001:db8:ab:100::/64 2001:db8:ab:100::1 eth1    0
0      0
2001:db8:ab:200::/64 2001:db8:ab:200::1 eth2    1
0      0
2001:db8:ab:300::/64 fe80::3333      eth2    1               254
2
2001:db8:ab:400::/64 fe80::3333      eth2    1               254
3
2001:db8:ab:500::/64 fe80::3333      eth2    1               254
4
2001:db8:ab:500::1/128 fe80::3333      eth2    1
254      4
2001:db8:ab:501::/64 fe80::3333      eth2    1               254
4
2001:db8:ab:501::1/128 fe80::3333      eth2    1
254      4
2001:db8:ab:600::/64 fe80::3333      eth2    1               254
3
```

```
2001:db8:ab:700::/64 fe80::3333      eth2    1                254
4

2001:db8:ab:700::1/128 fe80::3333      eth2    1
254      4

fe80::/64          fe80::1111      eth1    0                  0
0


root@router-xorp> show pim6 neighbors

Interface    DRpriority NeighborAddr    V Mode   Holdtime
Timeout

eth2                  1 fe80::3333      2 Sparse      105
76

                        2001:db8:ab:200::2


root@router-xorp> show ospf6 neighbor

  Address         Interface             State      ID
Pri  Dead

                eth2/eth2             Full      20.20.20.20
128    36

fe80::3333
```

Výpisy směrovače R3 (Xorp)

```
root@router-xorp:~# sysctl -a | grep ipv6 | grep forward

net.ipv6.conf.all.forwarding = 1

net.ipv6.conf.all.mc_forwarding = 1

net.ipv6.conf.default.forwarding = 1

net.ipv6.conf.default.mc_forwarding = 0

net.ipv6.conf.eth0.forwarding = 1

net.ipv6.conf.eth0.mc_forwarding = 0

net.ipv6.conf.eth1.forwarding = 1

net.ipv6.conf.eth1.mc_forwarding = 1

net.ipv6.conf.eth2.forwarding = 1

net.ipv6.conf.eth2.mc_forwarding = 1

net.ipv6.conf.eth3.forwarding = 1

net.ipv6.conf.eth3.mc_forwarding = 1

net.ipv6.conf.lo.forwarding = 1
```

```
net.ipv6.conf.lo.mc_forwarding = 0

net.ipv6.conf.pim6reg253.forwarding = 1

net.ipv6.conf.pim6reg253.mc_forwarding = 1


root@router-xorp:~# tcpdump -i eth1

13:53:17.573147 IP6 fe80::5555 > ff02::5: OSPFv3, Hello, length
40

13:53:19.481520 IP6 fe80::5555 > ff02::d: PIMv2, Hello, length
56

13:53:23.689891 IP6 fe80::4444 > ff02::d: PIMv2, Hello, length
56

13:53:24.425629 IP6 fe80::4444 > ff02::5: OSPFv3, Hello, length
40


root@router-xorp:~# tcpdump -i eth2

13:54:07.621443 IP6 fe80::8888 > ff02::5: OSPFv3, Hello, length
40

13:54:07.631382 IP6 fe80::6666 > ff02::5: OSPFv3, Hello, length
40

13:54:07.681057 IP6 fe80::8888 > ff02::d: PIMv2, Hello, length
56

13:54:17.621440 IP6 fe80::8888 > ff02::5: OSPFv3, Hello, length
40

13:54:17.631459 IP6 fe80::6666 > ff02::5: OSPFv3, Hello, length
40

13:54:19.283526 IP6 fe80::6666 > ff02::d: PIMv2, Hello, length


root@router-xorp:~# tcpdump -i eth3

13:56:07.577743 IP6 fe80::7777 > ff02::5: OSPFv3, Hello, length
40

13:56:08.777061 IP6 fe80::1235 > ff02::5: OSPFv3, Hello, length
40

13:56:14.816555 IP6 fe80::7777 > ff02::d: PIMv2, Hello, length
56

13:56:16.373558 IP6 fe80::1235 > ff02::d: PIMv2, Hello, length
56
```

```
root@router-xorp> show interfaces eth1
eth1/eth1: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet6 2001:db8:ab:300::2 prefixlen 64
        inet6 fe80::5555 prefixlen 64
        physical index 18
        ether 0:13:3b:a0:1:9e

root@router-xorp> show interfaces eth2
eth2/eth2: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet6 2001:db8:ab:400::1 prefixlen 64
        inet6 fe80::6666 prefixlen 64
        physical index 19
        ether 0:e0:4c:68:2:2e

root@router-xorp> show interfaces eth3
eth3/eth3: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet6 2001:db8:ab:600::1 prefixlen 64
        inet6 fe80::7777 prefixlen 64
        physical index 20
        ether 0:13:3b:a0:4:d4

root@router-xorp> show mfea6 interface
Interface    State    Vif/PifIndex Addr            Flags
eth1         UP               0/18 2001:db8:ab:300::2 MULTICAST
BROADCAST KERN_UP
eth2         UP               1/19 2001:db8:ab:400::1 MULTICAST
BROADCAST KERN_UP
eth3         UP               2/20 2001:db8:ab:600::1 MULTICAST
BROADCAST KERN_UP
register_vif UP               3/18 2001:db8:ab:300::2
PIM_REGISTER KERN_UP
```

```
root@router-xorp> show mld interface
Interface    State    Querier         Timeout Version Groups
eth1         UP       fe80::4444          164       2     13
eth2         UP       fe80::6666         None       2     13
eth3         UP       fe80::1235          196       2     13

root@router-xorp> show pim6 interface
Interface    State    Mode   V PIMstate Priority DRaddr
Neighbors
eth1         UP       Sparse 2 DR              1 fe80::5555
1
eth2         UP       Sparse 2 NotDR           1 fe80::8888
1
eth3         UP       Sparse 2 DR              1 fe80::7777
1
register_vif UP       Sparse 2 DR              1 fe80::5555
0

root@router-xorp> show pim6 rps
RP              Type      Pri Holdtime Timeout ActiveGroups
GroupPrefix
2001:db8:ab:300::2 static    192       -1      -1            2
ff00::/8

root@router-xorp> show pim6 join
Group           Source          RP              Flags
ff7e:240:2001:db8:ab:300:0:20 ::             2001:db8:ab:300::2
WC
    Upstream interface (RP):   register_vif
    Upstream MRIB next hop (RP): UNKNOWN
    Upstream RPF'(*,G):        UNKNOWN
    Upstream state:            Joined
    Join timer:                20
    Local receiver include WC: ....
    Joins RP:                  ....
```

```
    Joins WC:                   .O..
    Join state:                 .O..
    Prune state:                ....
    Prune pending state:        ....
    I am assert winner state:   ....
    I am assert loser state:    ....
    Assert winner WC:           ....
    Assert lost WC:             ....
    Assert tracking WC:         .O.O
    Could assert WC:            .O..
    I am DR:                    O.OO
    Immediate olist RP:         ....
    Immediate olist WC:         .O..
    Inherited olist SG:         .O..
    Inherited olist SG_RPT:     .O..
    PIM include WC:             .…

root@router-xorp> show pim6 neighbors
Interface    DRpriority NeighborAddr    V Mode   Holdtime
Timeout
eth1                  1 fe80::4444      2 Sparse      105
99
                        2001:db8:ab:300::1
eth2                  1 fe80::8888      2 Sparse      105
88
                        2001:db8:ab:400::2
eth3                  1 fe80::1235      2 Sparse      105
96
                        2001:db8:ab:600::2

root@router-xorp> show ospf6 neighbor
  Address         Interface             State      ID
Pri  Dead
                eth1/eth1               Full      20.20.20.20
128    33
```

```
fe80::4444
                eth2/eth2              Full      40.40.40.40
128    32
fe80::8888
                eth3/eth3              Full      50.50.50.50
128    33
fe80::1235
```

Výpisy směrovače R4 (Xorp)

```
root@router-xorp:~# sysctl -a | grep ipv6 | grep forward
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.all.mc_forwarding = 1
net.ipv6.conf.default.forwarding = 1
net.ipv6.conf.default.mc_forwarding = 0
net.ipv6.conf.eth0.forwarding = 1
net.ipv6.conf.eth0.mc_forwarding = 0
net.ipv6.conf.eth1.forwarding = 1
net.ipv6.conf.eth1.mc_forwarding = 1
net.ipv6.conf.eth2.forwarding = 1
net.ipv6.conf.eth2.mc_forwarding = 1
net.ipv6.conf.eth3.forwarding = 1
net.ipv6.conf.eth3.mc_forwarding = 1
net.ipv6.conf.lo.forwarding = 1
net.ipv6.conf.lo.mc_forwarding = 0
net.ipv6.conf.pim6reg253.forwarding = 1
net.ipv6.conf.pim6reg253.mc_forwarding = 1


root@router-xorp:~# tcpdump -i eth1
15:14:37.618054 IP6 fe80::8888 > ff02::5: OSPFv3, Hello, length
40
15:14:37.633307 IP6 fe80::6666 > ff02::5: OSPFv3, Hello, length
40
15:14:37.734980 IP6 fe80::8888 > ff02::d: PIMv2, Hello, length
56
15:14:49.547257 IP6 fe80::6666 > ff02::d: PIMv2, Hello, length
56
```

```
root@router-xorp:~# tcpdump -i eth2
15:15:02.939161 IP6 fe80::9999 > ff02::5: OSPFv3, Hello, length
36
15:15:07.528210 IP6 fe80::9999 > ff02::d: PIMv2, Hello, length
56
15:15:12.939067 IP6 fe80::9999 > ff02::5: OSPFv3, Hello, length
36


root@router-xorp:~# tcpdump -i eth3
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on eth3, link-type EN10MB (Ethernet), capture size
262144 bytes
15:16:06.975071 IP6 fe80::1234 > ff02::5: OSPFv3, Hello, length
36
15:16:10.487933 IP6 fe80::1234 > ff02::d: PIMv2, Hello, length
56
15:16:16.975067 IP6 fe80::1234 > ff02::5: OSPFv3, Hello, length
36


root@router-xorp> show interfaces eth1
eth1/eth1: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet6 2001:db8:ab:400::2 prefixlen 64
        inet6 fe80::8888 prefixlen 64
        physical index 6
        ether 0:e0:4c:68:2:64


root@router-xorp> show interfaces eth2
eth2/eth2: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet6 2001:db8:ab:500::1 prefixlen 64
        inet6 fe80::9999 prefixlen 64
        physical index 7
        ether 0:e0:4c:68:2:30
```

```
root@router-xorp> show interfaces eth3
eth3/eth3: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet6 2001:db8:ab:501::1 prefixlen 64
        inet6 fe80::1234 prefixlen 64
        physical index 8
        ether 0:e0:4c:68:3:1


root@router-xorp> show mfea6 interface
Interface    State    Vif/PifIndex Addr            Flags
eth1         UP               0/6 2001:db8:ab:400::2 MULTICAST
BROADCAST KERN_UP
eth2         UP               1/7 2001:db8:ab:500::1 MULTICAST
BROADCAST KERN_UP
eth3         UP               2/8 2001:db8:ab:501::1 MULTICAST
BROADCAST KERN_UP
register_vif UP               3/6 2001:db8:ab:400::2
PIM_REGISTER KERN_UP


root@router-xorp> show mld group
Interface    Group           Source          LastReported
Timeout V State
eth2         ff7e:240:2001:db8:ab:300:0:20 ::
fe80::2299:e45a:9bc2:cf2b     193 2     E
eth3         ff7e:240:2001:db8:ab:300:0:20 ::
fe80::ec35:3670:32cf:a469     190 2     E


root@router-xorp> show mld interface
Interface    State    Querier         Timeout Version Groups
eth1         UP       fe80::6666          227       2     13
eth2         UP       fe80::9999         None       2     14
eth3         UP       fe80::1234         None       2     14
```

```
root@router-xorp> show pim6 interface
Interface    State    Mode   V PIMstate Priority DRaddr
Neighbors
eth1         UP       Sparse 2 DR              1 fe80::8888
1
eth2         UP       Sparse 2 DR              1 fe80::9999
0
eth3         UP       Sparse 2 DR              1 fe80::1234
0
register_vif UP       Sparse 2 DR              1 fe80::8888
0


root@router-xorp> show pim6 join
Group           Source          RP              Flags
ff7e:240:2001:db8:ab:300:0:20 ::             2001:db8:ab:300::2
WC
    Upstream interface (RP):   eth1
    Upstream MRIB next hop (RP): fe80::6666
    Upstream RPF'(*,G):        fe80::6666
    Upstream state:            Joined
    Join timer:                35
    Local receiver include WC: .OO.
    Joins RP:                  ....
    Joins WC:                  ....
    Join state:                ....
    Prune state:               ....
    Prune pending state:       ....
    I am assert winner state:  ....
    I am assert loser state:   ....
    Assert winner WC:          ....
    Assert lost WC:            ....
    Assert tracking WC:        OOO.
    Could assert WC:           .OO.
    I am DR:                   OOOO
    Immediate olist RP:        ....
```

```
    Immediate olist WC:        .OO.
    Inherited olist SG:        .OO.
    Inherited olist SG_RPT:    .OO.
    PIM include WC:            .OO.

root@router-xorp> show pim6 neighbors
Interface    DRpriority NeighborAddr    V Mode   Holdtime
Timeout
eth1                 1 fe80::6666      2 Sparse      105
77
                         2001:db8:ab:400::1

root@router-xorp> show ospf6 neighbor
  Address         Interface             State      ID
Pri  Dead
                eth1/eth1              Full      30.30.30.30
128    36
fe80::6666
```

Příloha D: *Konfigurace směrovače R3 programu Xorpt pro IPv4*

Konfigurace směrovačů oproti IPv6 je rozdílná v tom, že obsahuje odlišné adresování a nastavení protokolů. Níže uvedená konfigurace se týká výhradně směrovače R3 pro základní ukázku. Zbylé konfigurační soubory nejsou přiloženy, jelikož jejich struktura je velmi podobná.

Konfigurace směrovače R3 (Xorp)

```
/* $XORP$ */
interfaces {
    interface eth1 {
    disable: false
        vif eth1 {
     disable: false
            address 10.30.30.2 {
                prefix-length: 24
         disable: false
            }
        }
    }
    interface eth2 {
    disable: false
        vif eth2 {
     disable: false
            address 10.40.40.1 {
                prefix-length: 24
         disable: false
            }
        }
    }
 interface eth3 {
    disable: false
        vif eth3 {
     disable: false
            address 10.70.70.1 {
                prefix-length: 24
```

```
          disable: false
            }
        }
    }
}

fea {
    unicast-forwarding4 {
        disable: false
    }
}
plumbing {
    mfea4 {
        disable: false
        interface eth1 {
            vif eth1 {
                disable: false
            }
        }
        interface eth2 {
            vif eth2 {
                disable: false
            }
        }
        interface eth3 {
            vif eth3 {
                disable: false
            }
        }
        interface register_vif {
            vif register_vif {
                disable: false
            }
```

```
        }
      traceoptions {
            flag all {
                  disable: false
            }
      }
       }
}
protocols {
    ospf4 {
        router-id: 30.30.30.30
        area 0.0.0.0 {
            interface eth1 {
                vif eth1 {
           address 10.30.30.2 {
           }
                }
            }
            interface eth2 {
                vif eth2 {
           address 10.40.40.1 {
           }
                    }
                }
            interface eth3 {
                vif eth3 {
           address 10.70.70.1 {
           }
                    }
                }
        }
    }
```

```
igmp {
    disable:false
    interface eth1 {
        vif eth1 {
            disable: false
            version: 3
        }
    }
    interface eth2 {
        vif eth2 {
            disable: false
            version: 3
        }
    }
    interface eth3 {
        vif eth3 {
            disable: false
            version: 3
        }
    }

 traceoptions {
 flag all {
      disable:false
 }
 }
}

pimsm4 {
    disable:false
    interface eth1 {
        vif eth1 {
```

```
            disable: false
        }
    }
    interface eth2 {
        vif eth2 {
            disable: false
        }
    }
    interface eth3 {
        vif eth3 {
            disable: false
        }
    }

    interface register_vif {
       vif register_vif {
       disable: false
       }
}

    static-rps {
        rp 10.30.30.2 {
            group-prefix 224.0.0.0/4 {
            }
        }
    }

traceoptions {
      flag all {
            disable: false
      }
       }
}
```

```
    fib2mrib {
        disable: false
    }
}
```

Příloha E: *Výpisy z testování programu Xorp pro IPv4*

Jednotlivé výpisy všech směrovačů jsou totožného charakteru, nicméně pro lepší znázornění jsou zde uvedeny výpisy směrovače R3 (Xorp) a příjemce multicastu.

Výpisy směrovače R3 (Xorp)

```
root@router-xorp:~# tcpdump -i eth1
12:28:11.950440 IP 10.10.10.10.43774 > 239.1.1.1.1234: UDP,
length 1316
12:28:11.950492 IP 10.10.10.10.43774 > 239.1.1.1.1234: UDP,
length 1316
12:28:11.950499 IP 10.10.10.10.43774 > 239.1.1.1.1234: UDP,
length 1316


root@router-xorp:~# tcpdump -i eth2
12:28:19.696808 IP 10.10.10.10.43774 > 239.1.1.1.1234: UDP,
length 1316
12:28:19.697950 IP 10.10.10.10.43774 > 239.1.1.1.1234: UDP,
length 1316
12:28:19.697963 IP 10.10.10.10.43774 > 239.1.1.1.1234: UDP,
length 1316


root@router-xorp:~# tcpdump -i eth3
12:28:24.185882 IP 10.10.10.10.43774 > 239.1.1.1.1234: UDP,
length 1316
12:28:24.185914 IP 10.10.10.10.43774 > 239.1.1.1.1234: UDP,
length 1316
12:28:24.185923 IP 10.10.10.10.43774 > 239.1.1.1.1234: UDP,
length 1316


root@router-xorp:~# sysctl -a | grep ipv4 | grep forward
net.ipv4.conf.all.forwarding = 1
net.ipv4.conf.all.mc_forwarding = 1
net.ipv4.conf.default.forwarding = 1
net.ipv4.conf.default.mc_forwarding = 0
net.ipv4.conf.eth0.forwarding = 1
net.ipv4.conf.eth0.mc_forwarding = 0
net.ipv4.conf.eth1.forwarding = 1
```

```
net.ipv4.conf.eth1.mc_forwarding = 1
net.ipv4.conf.eth2.forwarding = 1
net.ipv4.conf.eth2.mc_forwarding = 1
net.ipv4.conf.eth3.forwarding = 1
net.ipv4.conf.eth3.mc_forwarding = 1
net.ipv4.conf.lo.forwarding = 1
net.ipv4.conf.lo.mc_forwarding = 0
net.ipv4.conf.pimreg.forwarding = 1
net.ipv4.conf.pimreg.mc_forwarding = 1
net.ipv4.ip_forward = 1
net.ipv4.ip_forward_use_pmtu = 0

root@router-xorp> show interfaces eth1
eth1/eth1: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet 10.30.30.2 subnet 10.30.30.0/24 broadcast
10.30.30.255
        physical index 6
        ether 0:e0:4c:68:2:2b

root@router-xorp> show interfaces eth2
eth2/eth2: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet 10.40.40.1 subnet 10.40.40.0/24 broadcast
10.40.40.255
        physical index 7
        ether 0:e0:4c:68:2:1

root@router-xorp> show interfaces eth3
eth3/eth3: Flags:<ENABLED,BROADCAST,MULTICAST> mtu 1500 speed 1
Gbps
        inet 10.70.70.1 subnet 10.70.70.0/24 broadcast
10.70.70.255
        physical index 8
        ether 0:e0:4c:68:2:2e
```

```
root@router-xorp> show mfea interface
Interface    State    Vif/PifIndex Addr            Flags
eth1         UP                0/6 10.30.30.2      MULTICAST
BROADCAST KERN_UP
eth2         UP                1/7 10.40.40.1      MULTICAST
BROADCAST KERN_UP
eth3         UP                2/8 10.70.70.1      MULTICAST
BROADCAST KERN_UP
register_vif UP                3/6 10.30.30.2      PIM_REGISTER
KERN_UP


root@router-xorp> show mfea interface address
Interface    Addr            Subnet             Broadcast
P2Paddr
eth1         10.30.30.2      10.30.30.0/24      10.30.30.255
0.0.0.0
eth2         10.40.40.1      10.40.40.0/24      10.40.40.255
0.0.0.0
eth3         10.70.70.1      10.70.70.0/24      10.70.70.255
0.0.0.0
register_vif 10.30.30.2      10.30.30.2/32      10.30.30.2
0.0.0.0


root@router-xorp> show mfea dataflow
Group                                    Source
239.1.1.1                                10.10.10.10
  Measured(Start|Packets|Bytes) Type
Thresh(Interval|Packets|Bytes) Remain
  13180.81860|18635|?           <=     185.0|0|?
183.583593


root@router-xorp> show igmp interface
Interface    State    Querier         Timeout Version Groups
eth1         UP       10.30.30.1          163       3      5
eth2         UP       10.40.40.1         None       3      5
eth3         UP       10.70.70.1         None       3      5
```

```
root@router-xorp> show pim interface
Interface    State    Mode   V PIMstate Priority DRaddr
Neighbors
eth1         UP       Sparse 2 DR             1 10.30.30.2
1
eth2         UP       Sparse 2 NotDR          1 10.40.40.2
1
eth3         UP       Sparse 2 NotDR          1 10.70.70.2
1
register_vif UP       Sparse 2 DR             1 10.30.30.2
0


root@router-xorp> show pim join
Group           Source          RP              Flags
239.1.1.1       0.0.0.0         10.30.30.2      WC
    Upstream interface (RP):   register_vif
    Upstream MRIB next hop (RP): UNKNOWN
    Upstream RPF'(*,G):        UNKNOWN
    Upstream state:            Joined
    Join timer:                15
    Local receiver include WC: ....
    Joins RP:                  ....
    Joins WC:                  .OO.
    Join state:                .OO.
    Prune state:               ....
    Prune pending state:       ....
    I am assert winner state:  ....
    I am assert loser state:   ....
    Assert winner WC:          ....
    Assert lost WC:            ....
    Assert tracking WC:        .OOO
    Could assert WC:           .OO.
    I am DR:                   O..O
    Immediate olist RP:        ....
```

```
    Immediate olist WC:        .OO.
    Inherited olist SG:        .OO.
    Inherited olist SG_RPT:    .OO.
    PIM include WC:            ....
239.1.1.1       10.10.10.10     10.30.30.2      SG_RPT
    Upstream interface (S):    eth1
    Upstream interface (RP):   register_vif
    Upstream MRIB next hop (RP): UNKNOWN
    Upstream RPF'(S,G,rpt):    UNKNOWN
    Upstream state:            Pruned
    Override timer:            -1
    Local receiver include WC: ....
    Joins RP:                  ....
    Joins WC:                  .OO.
    Prunes SG_RPT:             ....
    Join state:                ....
    Prune state:               ....
    Prune pending state:       ....
    Prune tmp state:           ....
    Prune pending tmp state:   ....
    Assert winner WC:          ....
    Assert lost WC:            ....
    Assert lost SG_RPT:        ....
    Could assert WC:           .OO.
    Could assert SG:           .OO.
    I am DR:                   O..O
    Immediate olist RP:        ....
    Immediate olist WC:        .OO.
    Inherited olist SG:        .OO.
    Inherited olist SG_RPT:    .OO.
    PIM include WC:            ....
239.1.1.1       10.10.10.10     10.30.30.2      SG SPT
    Upstream interface (S):    eth1
```

```
Upstream interface (RP):   register_vif
Upstream MRIB next hop (RP): UNKNOWN
Upstream MRIB next hop (S):  10.30.30.1
Upstream RPF'(S,G):        10.30.30.1
Upstream state:            Joined
Join timer:                15
KAT(S,G) running:          true
Local receiver include WC: ....
Local receiver include SG: ....
Local receiver exclude SG: ....
Joins RP:                  ....
Joins WC:                  .OO.
Joins SG:                  ....
Join state:                ....
Prune state:               ....
Prune pending state:       ....
I am assert winner state:  ....
I am assert loser state:   ....
Assert winner WC:          ....
Assert winner SG:          ....
Assert lost WC:            ....
Assert lost SG:            ....
Assert lost SG_RPT:        ....
Assert tracking SG:        OOO.
Could assert WC:           .OO.
Could assert SG:           .OO.
I am DR:                   O..O
Immediate olist RP:        ....
Immediate olist WC:        .OO.
Immediate olist SG:        ....
Inherited olist SG:        .OO.
Inherited olist SG_RPT:    .OO.
PIM include WC:            ....
```

```
    PIM include SG:            ....
    PIM exclude SG:            ....
 --More-- (END)

root@router-xorp> show pim neighbors
Interface    DRpriority NeighborAddr    V Mode   Holdtime
Timeout
eth1                 1 10.30.30.1      2 Sparse      105
103
eth2                 1 10.40.40.2      2 Sparse      105
89
eth3                 1 10.70.70.2      2 Sparse      105
83

root@router-xorp> show pim rps
RP              Type      Pri Holdtime Timeout ActiveGroups
GroupPrefix
10.30.30.2      static    192       -1      -1            1
224.0.0.0/4

root@router-xorp> show ospf4 neighbor
  Address         Interface             State      ID
Pri  Dead
10.30.30.1       eth1/eth1             Full      20.20.20.20
128    36
10.40.40.2       eth2/eth2             Full      40.40.40.40
128    30
10.70.70.2       eth3/eth3             Full      50.50.50.50
128    34
```

Výpisy příjemce multicastu na adrese 10.50.50.10

```
root@pc14:~# tcpdump –i enx00ed4d680638
14:10:17.950749 IP 10.10.10.10.39904 > 239.1.1.1.1234: UDP,
length 1316
14:10:17.950777 IP 10.10.10.10.39904 > 239.1.1.1.1234: UDP,
length 1316
14:10:17.950780 IP 10.10.10.10.39904 > 239.1.1.1.1234: UDP,
length 1316
14:10:17.950783 IP 10.10.10.10.39904 > 239.1.1.1.1234: UDP,
length 1316
```

Příloha F: *Nastavení zařízení v testování programu MRD6*

Nastavení stream serveru

```
sudo –i
apt-get update
apt install vlc
service network-manager stop
ip link set enp1s0 down

ip a a 2001:db8:ab:100::10/64 dev nazevRozhrani
ip –6 route add 2001:db8:ab:200::/64 via 2001:db8:ab:100::1
ip –6 route add 2001:db8:ab:300::/64 via 2001:db8:ab:100::1
ip –6 route add 2001:db8:ab:301::/64 via 2001:db8:ab:100::1
sysctl –w net.ipv6.conf.all.forwarding=1
```

Nastavení příjemce na adrese 2001:db8:ab:300::10

```
sudo –i
apt-get update
apt install vlc
service network-manager stop
ip link set enp1s0 down

ip a a 2001:db8:ab:300::10/64 dev nazevRozhrani
ip –6 route add 2001:db8:ab:100::/64 via 2001:db8:ab:300::1
ip –6 route add 2001:db8:ab:200::/64 via 2001:db8:ab:300::1
sysctl –w net.ipv6.conf.all.forwarding=1
```

Nastavení příjemce na adrese 2001:db8:ab:301::10

```
sudo –i
apt-get update
apt install vlc
service network-manager stop
ip link set enp1s0 down

ip a a 2001:db8:ab:301::10/64 dev nazevRozhrani
ip –6 route add 2001:db8:ab:100::/64 via 2001:db8:ab:301::1
```

```
ip -6 route add 2001:db8:ab:200::/64 via 2001:db8:ab:301::1
sysctl -w net.ipv6.conf.all.forwarding=1
```

Nastavení směrovače R1 (MRD6)

```
sudo -i
apt-get update
apt install mrd6
service network-manager stop
ip link set enp1s0 down

ip a a 2001:db8:ab:100::1/64 dev nazevRozhrani
ip a a 2001:db8:ab:200::1/64 dev nazevRozhrani
ip -6 route add 2001:db8:ab:300::/64 via 2001:db8:ab:200::2
ip -6 route add 2001:db8:ab:301::/64 via 2001:db8:ab:200::2
sysctl -w net.ipv6.conf.all.forwarding=1

dpkg -L mrd6
cp /usr/share/doc/mrd6/examples/mrd.conf /etc/mrd6.conf
nano /etc/mrd6.conf

service mrd6 restart
service mrd6 status
```

Nastavení směrovače R2 (MRD6)

```
sudo -i
apt-get update
apt install mrd6
service network-manager stop
ip link set enp1s0 down

ip a a 2001:db8:ab:200::2/64 dev nazevRozhrani
ip a a 2001:db8:ab:300::1/64 dev nazevRozhrani
ip a a 2001:db8:ab:301::1/64 dev nazevRozhrani
ip -6 route add 2001:db8:ab:100::/64 via 2001:db8:ab:200::1
sysctl -w net.ipv6.conf.all.forwarding=1
```

```
dpkg -L mrd6
cp /usr/share/doc/mrd6/examples/mrd.conf /etc/mrd6.conf
nano /etc/mrd6.conf

service mrd6 restart
service mrd6 status
```

Příloha G: *Konfigurační soubory v testování programu MRD6*

Konfigurace směrovače R1 (MRD6)

```
/* MRD example configuration file */
log {
        /* Logs are controlled via the 'attach' method */
        /* syntax (one of):
           attach syslog [level]
           attach stderr [level]
           attach name filename [level]

           where level is one of:
                quiet, normal, verbose, debug or extradebug
         */
        attach stderr normal;
        attach default "mrd.log" debug;
}
load-module console;
load-module mld;
load-module pim;
console {
        /* Allow access from any host with admin/admin */
        /* allow-access admin admin any; */

        /* Command format: */
        /* allow-access [username [password [address mask]]]; */
}
/* Global pim variable configuration */
pim {
        /* we want to be a BSR candidate */
        enable bsr-candidate;
        /* we want to be a RP candidate */
}
```

```
/* Groups configuration */
groups {
        /* group mask */
        ff7e:240:2001:db8:ab:200::/96 {
                pim rp 2001:db8:ab:200::2;
        }
}
```

Konfigurace směrovače R2 (MRD6)

```
/* MRD example configuration file */
log {
        /* Logs are controlled via the 'attach' method */
        /* syntax (one of):
           attach syslog [level]
           attach stderr [level]
           attach name filename [level]

           where level is one of:
                quiet, normal, verbose, debug or extradebug
         */
        attach stderr normal;
        attach default "mrd.log" debug;
}
load-module console;
load-module mld;
load-module pim;

console {
        /* Allow access from any host with admin/admin */
        /* allow-access admin admin any; */

        /* Command format: */
        /* allow-access [username [password [address mask]]]; */
}
```

```
/* Global pim variable configuration */
pim {
        /* we want to be a BSR candidate */
        enable bsr-candidate;
        /* we want to be a RP candidate */
        enable rp-candidate;
}
/* Groups configuration */
groups {
        /* group mask */
        ff7e:240:2001:db8:ab:200::/96 {
                pim rp 2001:db8:ab:200::2;
        }
}
```

Příloha H: *Výpisy z testování programu MRD6*

Výpisy směrovače R1 (MRD6)

```
root@pc14:~# tcpdump -i enx00ec4d68068a
12:23:28.624536 IP6 2001:db8:ab:100::10.59664 >
ff7e:240:2001:db8:ab:200:0:20.1234: UDP, length 1316
12:23:28.624567 IP6 2001:db8:ab:100::10.59664 >
ff7e:240:2001:db8:ab:200:0:20.1234: UDP, length 1316
12:23:28.624573 IP6 2001:db8:ab:100::10.59664 >
ff7e:240:2001:db8:ab:200:0:20.1234: UDP, length 1316


root@pc14:~# tcpdump -i enx00ec4c680637
12:24:53.021289 IP6 pc14 > ff02::d: PIMv2, Hello, length 56
12:25:12.395009 IP6 fe80::1111 > ff02::d: PIMv2, Hello, length
56
12:25:15.748320 IP6 pc14 > 2001:db8:ab:200::2: PIMv2, Register,
length 52
12:25:15.749208 IP6 2001:db8:ab:200::2 > pc14: PIMv2, Register
Stop, length 42
12:25:23.019202 IP6 pc14 > ff02::d: PIMv2, Hello, length 56
12:25:28.019519 IP6 pc14 > ff02::d: PIMv2, Bootstrap, length 26
12:25:28.020551 IP6 fe80::1111 > ff02::d: PIMv2, Bootstrap,
length 26


root@pc14:~# sysctl -a | grep ipv6 | grep forward
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.all.mc_forwarding = 0
net.ipv6.conf.default.forwarding = 1
net.ipv6.conf.default.mc_forwarding = 0
net.ipv6.conf.enp1s0.forwarding = 1
net.ipv6.conf.enp1s0.mc_forwarding = 0
net.ipv6.conf.enx00ec4c680637.forwarding = 1
net.ipv6.conf.enx00ec4c680637.mc_forwarding = 0
net.ipv6.conf.enx00ec4d68068a.forwarding = 1
net.ipv6.conf.enx00ec4d68068a.mc_forwarding = 0
net.ipv6.conf.lo.forwarding = 1
```

```
net.ipv6.conf.lo.mc_forwarding = 0


root@pc14:~# mrd6sh show interface
Interface enp1s0 (2) is Down (Uptime: 17m 53s)
  Link-Local: (None)
Interface enx00ec4c680637 (4) is Up (Uptime: 17m 53s)
  Link-Local: fe80::3128:f8f3:8221:6fad/64
  Global: 2001:db8:ab:200::1/64 <PRIMARY>
  MLD, version 2
    Querier: fe80::1111 for 2m 54s
  PIM
    DR Priority: 1
    LAN Propagation Delay: 500ms Override Interval: 2500ms
    DR: self
    Neighbours:
      fe80::1111, 1m 42s
        DR-Priority: 1
        LAN Propagation Delay: 500ms Override Interval 2500ms
        Secundary-Addresses:
          2001:db8:ab:200::2
Interface enx00ec4d68068a (3) is Up (Uptime: 17m 53s)
  Link-Local: fe80::e8cf:4f9d:7aed:6d63/64
  Global: 2001:db8:ab:100::1/64 <PRIMARY>
  MLD, version 2
    Querier: self
  PIM
    DR Priority: 1
    LAN Propagation Delay: 500ms Override Interval: 2500ms
    DR: self
    Neighbours:
      (None)
Interface lo (1) is Up (Uptime: 17m 53s)
  Link-Local: ::1 <PRIMARY>
```

```
  MLD, version 2
    Querier: self

root@pc14:~# mrd6sh show group
Group ff7e:240:2001:db8:ab:200:0:20
  PIM
    Rendezvous-Point: 2001:db8:ab:200::2 [embedded]
    Sources:
      (2001:db8:ab:100::10), SPT, Active, Uptime: 18m 20s
        Register-Stop: 20s
        Input Interface: enx00ec4d68068a, Upstream: (Local)
```

Výpisy směrovače R2 (MRD6)

```
root@pc13:~# tcpdump -i enx00133ba00429
12:40:16.556766 IP6 2001:db8:ab:200::1 > pc13: PIMv2, Register,
length 52
12:40:16.556889 IP6 pc13 > 2001:db8:ab:200::1: PIMv2, Register
Stop, length 42
12:40:23.023668 IP6 fe80::3128:f8f3:8221:6fad > ff02::d: PIMv2,
Hello, length 56
12:40:28.028947 IP6 fe80::3128:f8f3:8221:6fad > ff02::d: PIMv2,
Bootstrap, length 26
12:40:28.029058 IP6 pc13 > ff02::d: PIMv2, Bootstrap, length 26
12:40:42.395730 IP6 pc13 > ff02::d: PIMv2, Hello, length 56
12:40:53.029142 IP6 fe80::3128:f8f3:8221:6fad > ff02::d: PIMv2,
Hello, length 56

root@pc13:~# tcpdump -i enx00133b9cbf3f
12:42:12.400575 IP6 pc13 > ff02::d: PIMv2, Hello, length 56
12:42:42.396442 IP6 pc13 > ff02::d: PIMv2, Hello, length 56

root@pc13:~# tcpdump -i enx00133ba004dc
12:43:42.394860 IP6 pc13 > ff02::d: PIMv2, Hello, length 56
12:44:12.391870 IP6 pc13 > ff02::d: PIMv2, Hello, length 56
```

```
root@pc13:~# sysctl -a | grep ipv6 | grep forward
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.all.mc_forwarding = 0
net.ipv6.conf.default.forwarding = 1
net.ipv6.conf.default.mc_forwarding = 0
net.ipv6.conf.enp1s0.forwarding = 1
net.ipv6.conf.enp1s0.mc_forwarding = 0
net.ipv6.conf.enx00133b9cbf3f.forwarding = 1
net.ipv6.conf.enx00133b9cbf3f.mc_forwarding = 0
net.ipv6.conf.enx00133ba00429.forwarding = 1
net.ipv6.conf.enx00133ba00429.mc_forwarding = 0
net.ipv6.conf.enx00133ba004dc.forwarding = 1
net.ipv6.conf.enx00133ba004dc.mc_forwarding = 0
net.ipv6.conf.lo.forwarding = 1
net.ipv6.conf.lo.mc_forwarding = 0

root@pc13:~# mrd6sh show interface
Interface enp1s0 (2) is Down (Uptime: 41m 40s)
  Link-Local: (None)
Interface enx00133b9cbf3f (6) is Up (Uptime: 41m 40s)
  Link-Local: fe80::213:3bff:fe9c:bf3f/64
  Global: 2001:db8:ab:300::1/64 <PRIMARY>
  MLD, version 2
    Querier: self
  PIM
    DR Priority: 1
    LAN Propagation Delay: 500ms Override Interval: 2500ms
    DR: self
    Neighbours:
      (None)
Interface enx00133ba00429 (3) is Up (Uptime: 41m 40s)
  Link-Local: fe80::1111/64
  Global: 2001:db8:ab:200::2/64 <PRIMARY>
```

```
  MLD, version 2
    Querier: self
  PIM
    DR Priority: 1
    LAN Propagation Delay: 500ms Override Interval: 2500ms
    DR: fe80::3128:f8f3:8221:6fad
    Neighbours:
      fe80::3128:f8f3:8221:6fad, 1m 16s
        DR-Priority: 1
        LAN Propagation Delay: 500ms Override Interval 2500ms
        Secundary-Addresses:
          2001:db8:ab:200::1
Interface enx00133ba004dc (8) is Up (Uptime: 41m 40s)
  Link-Local: fe80::213:3bff:fea0:4dc/64
  Global: 2001:db8:ab:301::1/64 <PRIMARY>
  MLD, version 2
    Querier: self
  PIM
    DR Priority: 1
    LAN Propagation Delay: 500ms Override Interval: 2500ms
    DR: self
    Neighbours:
      (None)
Interface lo (1) is Up (Uptime: 41m 40s)
  Link-Local: ::1 <PRIMARY>
  MLD, version 2
    Querier: self

root@pc13:~# mrd6sh show group
Group ff7e:240:2001:db8:ab:200:0:20
  PIM
    Rendezvous-Point: 2001:db8:ab:200::2 [embedded, self]
    Sources:
```

```
      (2001:db8:ab:100::10), Active, Uptime: 38m 54s
        Input Interface: enx00133ba00429, Upstream:
fe80::3128:f8f3:8221:6fad, No state
```

Příloha I: *Nastavení a konfigurace zařízení v testování programu SMCRoute*

Nastavení stream serveru

```
sudo –i
apt-get update
apt install vlc
ip link set enp1s0 down
service network-manager stop

ip a a 2001:db8:ab:100::10/64 dev nazevRozhrani
ip –6 route add 2001:db8:ab:200::/64 via 2001:db8:ab:100::1
ip –6 route add 2001:db8:ab:300::/64 via 2001:db8:ab:100::1
sysctl –w net.ipv6.conf.all.forwarding=1
```

Nastavení příjemce na adrese 2001:db8:ab:300::10

```
sudo –i
apt-get update
apt install vlc
ip link set enp1s0 down
service network-manager stop

ip a a 2001:db8:ab:300::10/64 dev nazevRozhrani
ip –6 route add 2001:db8:ab:100::/64 via 2001:db8:ab:300::1
ip –6 route add 2001:db8:ab:200::/64 via 2001:db8:ab:300::1
sysctl –w net.ipv6.conf.all.forwarding=1
```

Nastavení příjemce na adrese 2001:db8:ab:301::10

```
sudo –i
apt-get update
apt install vlc
ip link set enp1s0 down
service network-manager stop

ip a a 2001:db8:ab:301::10/64 dev nazevRozhrani
ip –6 route add 2001:db8:ab:100::/64 via 2001:db8:ab:301::1
ip –6 route add 2001:db8:ab:200::/64 via 2001:db8:ab:301::1
```

```
sysctl -w net.ipv6.conf.all.forwarding=1
```

Nastavení a konfigurace směrovače R1 (SMCRoute)

```
sudo -i
apt-get update
apt install smcroute
ip link set enp1s0 down
service network-manager stop

ip a a 2001:db8:ab:100::1/64 dev nazevRozhrani
ip a a 2001:db8:ab:200::1/64 dev nazevRozhrani
ip -6 route add 2001:db8:ab:300::/64 via 2001:db8:ab:200::2
sysctl -w net.ipv6.conf.all.forwarding=1

dpkg -L smcroute
cp /usr/share/doc/smcroute/examples/smcroute.conf
/etc/smcroute.conf
nano /etc/smcroute.conf

mgroup from "nazev rozhrani(2001:db8:ab:100::1)" group ff3e::20
mgroup from "nazev rozhrani(2001:db8:ab:100::1)" group ff3e::20
source 2001:db8:ab:100::10 to "nazev
rozhrani(2001:db8:ab:200::1)"

service smcroute restart
service smcroute status
```

Nastavení a konfigurace směrovače R2 (SMCRoute)

```
sudo -i
apt-get update
apt install smcroute
ip link set enp1s0 down
service network-manager stop

ip a a 2001:db8:ab:200::2/64 dev nazevRozhrani
ip a a 2001:db8:ab:300::1/64 dev nazevRozhrani
```

```
ip -6 route add 2001:db8:ab:100::/64 via 2001:db8:ab:200::1
sysctl -w net.ipv6.conf.all.forwarding=1


dpkg -L smcroute
cp /usr/share/doc/smcroute/examples/smcroute.conf
/etc/smcroute.conf
nano /etc/smcroute.conf


mgroup from "nazev rozhrani(2001:db8:ab:200::2)" group ff3e::20
mgroup from "nazev rozhrani(2001:db8:ab:200::2)" group ff3e::20
source 2001:db8:ab:100::10 to "nazev
rozhrani(2001:db8:ab:300::1)" "nazev
rozhrani(2001:db8:ab:301::1)"


service smcroute restart
service smcroute status
```

Příloha J: *Skripty pro testování programu SMCRoute*

Skript stream serveru

```
#!/bin/bash
echo Provadim update repozitaru a instalaci programu VLC.
sleep 3
apt-get update
apt-get install -y vlc
echo Nastavuji IPv6 adresu stroje a site.
sleep 3
service network-manager stop
ip link set enp1s0 down
ip link set nazevRozhrani up
ip addr add 2001:db8:ab:100::10/64 dev nazevRozhrani
ip -6 route add 2001:db8:ab:200::/64 via 2001:db8:ab:100::1
ip -6 route add 2001:db8:ab:300::/64 via 2001:db8:ab:100::1
ip -6 route add 2001:db8:ab:301::/64 via 2001:db8:ab:100::1
((count = 10))
while [[ $count -ne 0 ]] ; do
    ping -c 3 2001:db8:ab:100::1
    rc=$?
    if [[ $rc -eq 0 ]] ; then
        ((count = 1))
    fi
    ((count = count - 1))
done
if [[ $rc -eq 0 ]] ; then
    echo "Rozhrani je pripojeno."
else
    echo "Rozhrani neni pripojeno."
fi
```

Skript příjemců multicastu

```
#!/bin/bash
echo Provadim update repozitaru a instalaci programu VLC.
sleep 3
apt-get update
apt-get install -y vlc
echo Nastavuji IPv6 adresu stroje a site.
sleep 3
service network-manager stop
ip link set enp1s0 down
ip link set nazevRozhrani up
ip addr add IPv6_adresa/64 dev nazevRozhrani
ip -6 route add 2001:db8:ab:100::/64 via IPv6_adresa
ip -6 route add 2001:db8:ab:200::/64 via IPv6_adresa

((count = 10))
while [[ $count -ne 0 ]] ; do
    ping -c 3 IPv6_adresa
    rc=$?
    if [[ $rc -eq 0 ]] ; then
        ((count = 1))
    fi
    ((count = count - 1))
done
if [[ $rc -eq 0 ]] ; then
    echo "Rozhrani je pripojeno."
else
    echo "Rozhrani neni pripojeno."
fi
```

Skript směrovačů

```
#!/bin/bash
echo Provadim update repozitaru a instalaci programu SMCRoute.
sleep 3
apt-get update
apt install -y smcroute
echo Nastavuji IPv6 adresy rozhranim a routy.
sleep 3
service network-manager stop
ip link set enp1s0 down
ip link set nazevRozhrani up
ip link set nazevRozhrani up
ip addr add IPv6_adresa/64 dev nazevRozhrani
ip addr add IPv6_adresa /64 dev nazevRozhrani
ip -6 route add IPv6_adresa/64 via IPv6_adresa
ip -6 route add IPv6_adresa/64 via IPv6_adresa
sysctl -w net.ipv6.conf.all.forwarding=1
cp /usr/share/doc/smcroute/examples/smcroute.conf
/etc/smcroute.conf
echo Nyni je potreba upravit soubor /etc/smcroute.conf dle
potreby.
```

Příloha K: *Výpisy z testování programu SMCRoute*

Výpisy stream serveru

```
root@pc1:~# tcpdump –i enx00eb4e68047b
14:27:11.729844 IP6 pc1.36430 > ff3e::20.1234: UDP, length 1316
14:27:11.729936 IP6 pc1.36430 > ff3e::20.1234: UDP, length 1316
14:27:11.730005 IP6 pc1.36430 > ff3e::20.1234: UDP, length 1316
```

Výpisy příjemců

```
root@pc4:~# tcpdump –i enx00e34c680899
14:56:27.491558 IP6 2001:db8:ab:100::10.57362 > ff3e::20.1234:
UDP, length 1316
14:56:27.491599 IP6 2001:db8:ab:100::10.57362 > ff3e::20.1234:
UDP, length 1316
14:56:27.491603 IP6 2001:db8:ab:100::10.57362 > ff3e::20.1234:
UDP, length 1316
```

```
root@pc5:~# tcpdump –i enx00ed4d680638
15:02:30.116907 IP6 2001:db8:ab:100::10.57362 > ff3e::20.1234:
UDP, length 1316
15:02:30.116931 IP6 2001:db8:ab:100::10.57362 > ff3e::20.1234:
UDP, length 1316
15:02:30.116936 IP6 2001:db8:ab:100::10.57362 > ff3e::20.1234:
UDP, length 1316
```

Výpisy směrovače R1 (SMCRoute)

```
root@pc2:~# tcpdump –i enx00eb4e680462
14:35:59.459542 IP6 2001:db8:ab:100::10.37615 > ff3e::20.1234:
UDP, length 1316
14:35:59.459583 IP6 2001:db8:ab:100::10.37615 > ff3e::20.1234:
UDP, length 1316
14:35:59.460663 IP6 2001:db8:ab:100::10.37615 > ff3e::20.1234:
UDP, length 1316
```

```
root@pc2:~# tcpdump –i enx00ec4d680604
14:37:18.171336 IP6 2001:db8:ab:100::10.37615 > ff3e::20.1234:
UDP, length 1316
```

```
14:37:18.171366 IP6 2001:db8:ab:100::10.37615 > ff3e::20.1234:
UDP, length 1316
14:37:18.172836 IP6 2001:db8:ab:100::10.37615 > ff3e::20.1234:
UDP, length 1316


root@pc2:~# sysctl -a | grep ipv6 | grep forward
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.all.mc_forwarding = 1
net.ipv6.conf.default.forwarding = 1
net.ipv6.conf.default.mc_forwarding = 0
net.ipv6.conf.enp1s0.forwarding = 1
net.ipv6.conf.enp1s0.mc_forwarding = 0
net.ipv6.conf.enx00eb4e680462.forwarding = 1
net.ipv6.conf.enx00eb4e680462.mc_forwarding = 1
net.ipv6.conf.enx00ec4d680604.forwarding = 1
net.ipv6.conf.enx00ec4d680604.mc_forwarding = 1
net.ipv6.conf.lo.forwarding = 1
net.ipv6.conf.lo.mc_forwarding = 0
```